AD-A259 638

TACTICAL WEAPON
GACIAC
GUIDANCE & CONTROL
INFORMATION ANALYSIS CENTER

GACIAC PR 92-02

PROCEEDINGS OF THE GOVERNMENT NEURAL NETWORK APPLICATIONS WORKSHOP - Volume 1

24-26 August 1992

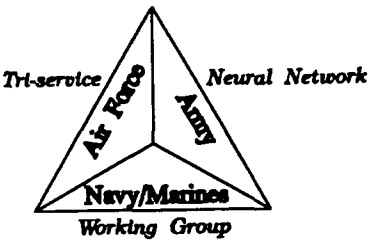
DTIC
ELECTE
JAN1 3 1993
S C D

Conducted at:

Wright-Patterson AFB, Ohio

Sponsored by:

Tri-Service Neural Network Working Group



Approved for public release; distribution is unlimited.

DEFENSE TECHNICAL INFORMATION CENTER
9300772
13308

Published by GACIAC
IIT Research Institute
10 West 35th Street
Chicago, Illinois 60616-3799

93 1 01 009

NOTICES

Proceedings. This proceedings has been published by the Tactical Weapon Guidance and Control Information Analysis Center (GACIAC) as a service to the defense community. GACIAC is a DoD Information Analysis Center, administered by the Defense Technical Information Center, operated by IIT Research Institute under Contract No. DLA-900-86-C-0022. GACIAC is funded by DTIC, DARPA and U.S. Army, U.S. Navy, and U.S. Air Force Laboratories/Controlling Activities having an interest in tactical weapon guidance and control. The Director of GACIAC is Dr. Robert J. Heaston. The Contracting Officer is Ms. Cheryl Montoney, DESC, Dayton, Ohio. The Contracting Officer's Technical Representative is Mr. Chalmer D. George, and the Alternate Representative is Mr. H.C. Race, AMC Smart Weapons Management Office, ATTN: AMSMI-SW, Redstone Arsenal, Alabama 35898-5222.

Reproduction. Permission to reproduce any material contained in this document must be requested from, and approved in writing by the Air Force Office of Scientific Research, Mathematics Department (AFOSR/NM), Bolling AFB, DC 20332-6448. This document is only available from GACIAC, IIT Research Institute, 10 West 35th Street, Chicago, Illinois 60616-3799. Copies are available to Conference attendees, Government agencies, and GACIAC industrial subscribers.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information including suggestions for reducing the burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1992	Proceedings; 24-26 August 1992

4. TITLE AND SUBTITLE
Proceedings of the Government Neural Network Applications Workshop - Volume 1

5. FUNDING NUMBERS
DoD-DLA900-86-C-0022
PE 65802
PR 1.0

6. AUTHOR(S)
Various; Capt. Steve Suddarth, General Chairman;
Maj. Steven K. Rogers, Program Chairman.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
IIT Research Institute/GACIAC
10 West 35th Street
Chicago, IL 60616-3799

8. PERFORMING ORGANIZATION REPORT NUMBER
GACIAC PR 92-02
Volume 1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Director
Air Force Office of Scientific Research
Mathematics Department (AFOSR/NM)
Bolling AFB, DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
This document is available only from GACIAC, IIT Research Institute, 10 West 35th Street, Chicago, IL 60616-3799. (410948)

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This volume contains the proceedings of the Government Neural Networks Application Workshop held 24-26 August 1992 at the Hope Hotel, Wright-Patterson AFB, OH. The Tri-Service Neural Network working group organizes this meeting annually to highlight achievements in applying neural networks to government problems.

Papers presented addressed the following applications: manufacturing/logistics, adaptive control systems, robotics, machine vision and ATR, radar and signal processing, and performance and system integration. Volume 1 contains unclassified/Unlimited papers. Volume 2 contains the unclassified/limited papers.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Networks, Neural networks, Learning systems, Target detection, Automatic target recognition, Image processing, Signal processing, Radar, Synthetic aperture radar, Sonar, Nondestructive testing, Control systems, Machine vision, Symposia.	152
	16. PRICE CODE: $100

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED/UNLIMITED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

THIS PAGE IS INTENTIONALLY BLANK

GACIAC PR 92-02 August 1992

PROCEEDINGS OF THE GOVERNMENT NEURAL NETWORK APPLICATIONS WORKSHOP - Volume 1

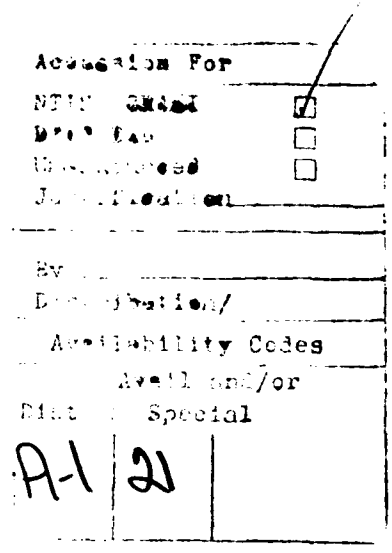
24-26 August 1992

Conducted at:

Wright-Patterson AFB, Ohio

Sponsored by:

Tri-Service Neural Network Working Group



Distribution:

Approved for public release; distribution is unlimited.

Published by GACIAC
IIT Research Institute
10 West 35th Street
Chicago, Illinois 60616-3799

GACIAC - A DoD Information Analysis Center
Operated by IIT Research Institute, 10 West 35th Street, Chicago, Illinois 60616-3799
DoD Technical Sponsor-Joint Service Guidance and Control Committee
Members from OSD, Army, Navy, Air Force and DARPA

Acknowledgments

The following people provided great assistance in preparing for this meeting:

Program Chair: Maj. Steve Rogers

Program Committee: Dr. Clifford Lau
Dr. Tim Ross
Capt. Steve Suddarth

Tutorials: Maj. Steve Rogers
Capt. Dennis Ruck

Proceedings / Publications / Registration:
Ms. Mindy Tumarkin
Ms. Toni Cavalieri

Local Arrangements / Tutorial Preparation:
Ms. Juanita Timm

Foreword

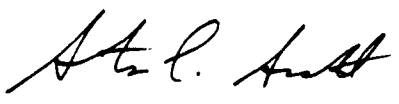
Neural networks are of great interest for solving problems in computation that have previously been considered difficult. Since the late 1970's, interest in neural computation has grown with the advent of new general-purpose network topologies and learning algorithms. During this process, government agencies, laboratories and contractors have played key roles in the development of theories and analyses that will be used world-wide for a long time to come. The reason for this contribution has generally been the government's need to develop advanced computer applications. While embedded weapon system applications have certainly been a celebrated area of the field, the government's needs cover a wide range of fields, including logistics, management, and medicine.

This volume contains the proceedings of the Government Neural Networks Application Workshop (GNN92) held 24-26 August 1992 at the Hope Hotel, Wright-Patterson AFB, OH. The Tri-Service Neural Network Working Group organizes this meeting annually to highlight achievements in applying neural networks to government problems. By emphasizing applications over theory, this meeting serves unique role in the neural network research community for enhancing team interaction within government, academe and industry. The meeting has also proved useful for providing feedback to theoretical researchers on the relevance and usefulness of their results.

This year, the conference noted particular advances in the following areas:

- Manufacturing/Logistics
- Adaptive control
- Robotics
- Automatic target recognition
- Radar/signal processing systems

Other papers discussed means of improving neural network performance and system integration. It is our hope that this meeting will enhance communication and serve as a springboard for future collaboration among researchers who are keenly interested in applications.



Capt. Steve Suddarth
GNN92 Chair

Government Neural Network Applications Workshop - GNN 92

Hope Hotel, WP AFB, Dayton, Ohio - August 24-26, 1992

AGENDA

24 August 1992

0900-1200
1300-2000 **Registration**

Tutorial Day ***"Introduction to Biological and Artificial Neural Networks"***

This Tutorial will be led by Maj. Steve Rogers of AFIT. It will provide an introduction for potential users to understand the rest of the meeting and to size up their application needs against available technologies. Tutorial registration will be $7. The fee provides for the meeting room, a continental breakfast, and refreshments.

0830-0945 Tutorial 1: Biological Foundation and Historical Perspective of Neural Computation

0945-1000 Break

0945-1145 Tutorial 2: Supervised Learning Techniques

1145-1300 Break for Lunch (not provided)

1300-1445 Tutorial 3: Unsupervised Learning

1445-1500 Break

1500-1645 Tutorial 4: Applications (with emphasis on government uses)

25 Aug - Main Meeting (Day 1)

0730-1200
1300-1800 **Registration**

0800-0940 **Export-Controlled I** *(Must have authorization & I.D. to attend)*

Sonar Target Detection Using Analog Neural Networks
P. Mukai, M. Busa - Draper Laboratory

An Efficient Hierarchical Neural Network Architecture for Discriminating Time-Varying Underwater Signals
D. J. Adams, S. Clary - Loral Defense Systems, Akron;
Y.-H. Pao, T.L. Hemminger, P. Yip - Case Western Reserve University

Performance Analysis of the ART-2 Network in a SAR Target Recognition System
S.M. Verbout, L.M. Novak - MIT Lincoln Laboratory

Neural Networks for the Anti-Radar Missile Application
A. Penz, P. Thrift, A. Katz - Texas Instruments

0940-0955 Break

0955-1110 **Export-Controlled II** *(Must have authorization and I.D. to attend)*

Neural System for Learning and Recognition of Targets from Spotlight Mode SAR Sequences
A.M. Waxman, A.M. Bernardon, M. Seibert, D. Fay - MIT Lincoln Laboratory

The Missile-Borne Integrated Neural Network Demonstration Program
M.L. Mumford, D.K. Andes, L.R. Kern - Naval Weapons Center

The Neocognitron for Viewpoint Invariant Automatic Target Recognition
J. McKinstry, T. Ruff, A. Ott - Logicon, Inc.

1110-1120 **Break**

1120-1235 **Export Controlled III** *(must have authorization and I.D. to attend)*

ELINT Signal Classification Using Neural Networks
E.W. Davis, H.K. Chung - E-Systems

Hardware Implementation of an ART 2 Neural Network for Intrapulse Analysis
M.T. Kopp, J.H. Schlag, K.L. Schlag - Georgia Tech; B. Andrews, Wright Laboratory

The Design of a Two-Layered, One-Step-Learning Perceptron
C.-L.J. Hu - Southern Illinois University

1235-1345 **Lunch**

1345-1500 **Manufacturing/Logistics I**

Neural Network Application for Process Modelling in the Navy Weldexcell Off-Line Engineering/Planning Workstation
X. Xu, J.E. Jones - Colorado School of Mines

Neural Network Analysis of Nondestructive Evaluation Patterns
W.E. Bond - McDonnell Douglas Research Laboratories;
D.C. St. Clair - University of Missouri-Rolla;
M.M. Amirfathi - Douglas Aircraft;
C.J. Merz, S. Aylward - McDonnell Aircraft

An Application of Neural Network Techniques to Statistical Process Control Chart Interpretation
M. Shewhart - Center for Supportability and Technology Insertion

1500-1515 **Break**

1515-1605 **Manufacturing/Logistics II**

Cell Management Using Neural Network Approaches
D.-S. Yang - University of Illinois at Urbana-Champaign;
J. Garret - Carnegie-Melon University;
D. Smith Shaw - U.S. Army Construction Engineering Research Laboratory

A Neural Network Approach to Fault Isolation in Units Under Test (UUTs)
P.V. Hayes - IIT Defense, R.T. Rue, S.I. Sayegh - Purdue University (Fort Wayne)

1605-1620 **Break**

1620-1800 **Session 5: Implementation Issues**

Applications of Computer Algebra Systems to Neural Networks
S.I. Sayegh - Purdue University (Fort Wayne)

Membrane Equation and Implementation of Neural Networks
B. Nabet - Drexel University

Oscillatory Neural Networks on Multiple Limit-Cycle Self-Oscillators
Y. A. Saet

Design and Implementation of the ACU: An Expandable ANN Building Block
R. J. Schalkoff, K.F. Poole, R.E. Owens, J.N. Gowdy, A.E. Turner - Clemson University

1800-1900 Cocktail hour - Demonstrations

1900-2030 Dinner

26 Aug - Main Meeting (Day 2)

0800-0915 **Controls I**

Dynamically Reconfigurable Neural Networks for Landmark Recognition
R.C Luo, H. Potlapalli - North Carolina State University;
D.W. Hislop - U.S. Army Research Office

Adaptation and Learning in Control Systems
W. Baker, J. Farrell - Draper Laboratory

A Neural-Net System for Real-Time Learning of Control Concepts
Y.-H. Pao - Case Western Reserve University

0915-0930 **Break**

0930-1110 **Vision/ATR I**

A Low Cost Foveal Vision System
R. Hecht-Nielsen, Y. T. Zhou - HNC, Inc.

Neural Network Data Fusion for Classification of Earth Surface from SSMI Measurements
Y.M. Fleming Lure, Y.S.P. Chiou, H.Y.M. Yeh - Caelum Research Corporation;
N.C. Grody - National Oceanic and Atmospheric Administration

Classification of Multispectral Data: A Comparison between Neural Network and Classical Techniques
J. H. Seldin, J.N. Cederquist - ERIM

Track Before Detect Using Model Based Neural Network
L. Perlovsky - Nichols Research Corporation

1110-1210 Lunch

1210-1300 Radar/Signal Processing

An Architected Neural Network for Contact State Estimations
C. DeAngelis - Naval Undersea Warfare Center Division;
R.W. Green - University of Massachusetts Dartmouth

1300-1315 Break

1315-1430 **Vision/ATR II**

Novel Optical Neural Network for Target Detection and Classification
J.N. Cederquist, D.K. Angell, J.H. Seldin - ERIM; K.A. Weigand, Wright Laboratory

The SAHTIRN System for ATR
W.P. Lincoln, W. Tackett, C. Daniell, T. Baraghimian - Hughes Aircraft Company

Neural Networks for Classifying Image Textures
S.H. Lane, J.C. Pearson, R. Sverdlove - David Sarnoff Research Center

1430-1445 Break

1445-1535 **Vision/ATR III**

Retrieving Atmospheric Temperature and Moisture Profiles from DMSP Sounder Data with NN
C.T. Butler - Physical Sciences, Inc.;
R. v. Z. Meredith - Research Support Instruments, Inc.

1535-1550 Closing Remarks

Alternate Paper - *"An Alternative Approach, Using Goal Program Before the Training and Configuration of Artificial Neural Network"* by Prof. James P. Ignizio

THIS PAGE IS INTENTIONALLY BLANK

Government Neural Network Applications Workshop

Volume 1
TABLE OF CONTENTS

Page

Manufacturing/Logistics I

"Neural Network Application for Process Modelling in the Navy WELDEXCELL Off-Line Engineering/Planning Workstation" 1
X. Xu, and J.E. Jones

"Using Neural Networks to Analyze Nondestructive Evaluation Patterns" 7
W.E. Bond, and D.C. St. Clair

"An Application of Neural Network Techniques to Statistical Process Control Chart Interpretation" 13
M. Shewhart

Manufacturing/Logistics II

"Cell Management Using Neural Network Approaches" 19
D.-S. Yang, J.H. Garrett, Jr., and D. Smith Shaw

"A Neural Network Approach to Fault Isolation in Units Under Test (UUTs)" 25
P.V. Hayes, R.T. Rue, and S.I. Sayegh

Session 5: Implementation Issues

"Applications of Computer Algebra Systems to Neural Networks" 31
S.I. Sayegh

"Membrane Equation and Implementation of Neural Networks" 37
B. Nabet

"Oscillatory Neural Networks on Multiple Limit-Cycle Self-Oscillators" 43
Y.A. Saet

"Design and Implementation of the ACU: An Expandable ANN Building Block" 49
R.J. Schalkoff, K.F. Poole, R. Singh, R.E. Owens, J.N. Gowdy, and A.E. Turner

Controls I

"Dynamically Reconfigureable Neural Network for Landmark Recognition" 55
R.C Luo, H. Potlapalli, and D.W. Hislop

"Adaptation and Learning in Control Systems: Application in Flight Control" 61
W.L. Baker, and P.J. Millington

Vision/ATR I

"A Low Cost Foveal Vision System" 67
R. Hecht-Nielsen, and Y. T. Zhou

"Neural Network Data Fusion for Classification of Earth Surface from SSMI Measurements" 73
Y.M. Fleming Lure, N.C. Grody, Y.S.P. Chiou, and H.Y.M. Yeh

"Classification of Multispectral Data: A Comparison between Neural Network and Classical Techniques" 79
J. H. Seldin, and J.N. Cederquist

"Track Before Detect Using Model Based Neural Network" 85
L. Perlovsky

Radar/Signal Processing

"An Architected Neural Network for Contact State Estimation: A Comparison to the Cramer-Rao Lower Bound" 89
C.M. DeAngelis, K.J. Ross, and R.W. Green

"Classification of Long Signal Segments 95
B. de Vries, R. Sverdlove, S. Markel, J. Pearson, and S.Y. Kung

Vision/ATR II

"Novel Optical Neural Network for Target Detection and Classification" 101
J.N. Cederquist, D.K. Angell, J.H. Seldin, and K.A. Weigand

"The SAHTIRN System for ATR" 105
W.P. Lincoln, C.E. Daniell, W.A. Tackett, and G.A. Baraghimian

"Neural Networks for Classifying Image Textures" 111
S.H. Lane, J.C. Pearson, and R. Sverdlove

Vision/ATR III

"Retrieving Atmospheric Temperature and Moisture Profiles from DMSP Sounder Data with A Neural Network" 123
C.T. Butler, and R.v.Z. Meredith

"Clustering with Hopfield Neural Networks" 129
B. Kamgar-Parsi, and B. Kamgar-Parsi

Order Form for Proceedings of this Workshop 135

Order Form to receive GACIAC Bulletin 137

Presented at the Government Neural Network Applications Workshop

24-26 August 1992 GACIAC PR 92-02

NEURAL NETWORK APPLICATION FOR PROCESS MODELLING IN THE NAVY *WELDEXCELL* OFF-LINE ENGINEERING/PLANNING WORKSTATION

1 July 1992

Xiaoshu Xu
American Welding Institute
10628 Dutchtown Road
Knoxville, TN 37932

Jerald E. Jones
AWI Professor, Center for Artificial Intelligence
Colorado School of Mines
Golden, CO 80401

ABSTRACT

Artificial neural systems (ANS), also known as neural networks, are an attempt to develop computer systems that emulate the neural reasoning behavior of biological neural systems (e.g. the human brain). As such, they are loosely based on biological neural networks. The ANS consists of a series of nodes (neurons) and weighted connections (axons) that, when presented with a specific input pattern, can associate specific output patterns. It is essentially a highly complex, non-linear, mathematical relationship or transform. These constructs have two significant properties that have proven useful to the authors in process modelling: noise tolerance and complex pattern recognition. Specifically, the authors have developed a new network learning algorithm that has resulted in the successful application of ANS's to development of models of highly complex processes. The Navy's WELDEXCELL off-line planning workstation for automated and robotic welding utilizes such a process model in setting the process control parameters. This paper describes, briefly, the WELDEXCELL System, and discusses the WELDBEAD neural network process model that is being integrated into that workstation software.

Keywords: Artificial Neural System, Neural Network, WELDEXCELL, Modelling, Welding, Learning, Training, Delta-Activity, Application

1.0 INTRODUCTION: ARTIFICIAL NEURAL SYSTEMS

It is assumed that the reader has a basic working knowledge of Artificial Neural Systems (ANS). These systems which are loosely based on biological neural networks offer a computer technology that is a useful tool in process modelling. The ANS consists of a series of nodes (neurons) and weighted connections (axons). As with a biological neural network, the assignment of the values of the weights and the size and configuration of the network are the key to a successful net. Unfortunately, we have only begun to understand the inner workings of these constructs. Consequently, relatively crude tools are currently employed to develop working networks.

Typically, the approach to model development is to first develop a thorough understanding of the underlying basic scientific or engineering principles of the process. Then, a model is developed that is based on mathematical relationships inherent in the process. The principal drawback with this approach is the time and effort required to develop an understanding of these basic scientific or engineering relationships. Depending on the complexity of the problem, it can take many years and an extensive research program to develop the relationships. Often, instead, a number of simplifying assumptions are made and an approximate

model is developed. That approach, while being an expedient method for developing an approximate model that could be useful, provides only a theoretical approximation that may not be valid for the actual problem application.

The artificial neural system, when a network can be found to solve the problem, provides an accurate empirical model of the process. Neural networks are empirical bases systems that, when presented with a specific input pattern, can associate specific output patterns. It is, essentially, a highly complex, non-linear, mathematical relationship or transform. However; it is not necessary for the developer of such a system to understand the basic underlying principles of a process in order to develop a highly accurate ANS based model of the process. Thus, in this way it is quite different from other mathematical modelling approaches.

1.1 THE DELTA-ACTIVITY™ NETWORK

A new method was developed by the authors for training neural networks that has been shown to overcome all of the known problems with the gradient descent type methods for neural network development, while maintaining the inherent stability and known network development capabilities of those methods. This network was developed through the use of a thermodynamic model of the network operation which included both the delta energy and the activity or kinetics of the network. The technique, known as the Delta-Activity Network (D-A Net), has been used on several applications ranging from manufacturing process modelling, to high speed signal processing for electronic warfare applications, to underwater acoustic signal classification.

The D-A Net has achieved learning rates as high as 1000 times that of the back propagation method while also preventing the network from falling into learning instabilities[1]. In addition, the system configures itself dynamically during the learning process and so it can produce a near optimum network size for operation, often much smaller than the network needed to "learn" the problem. By the combination of dynamic self-configuration and learning instability avoidance, an operating network is virtually guaranteed for all process modelling problems. Utilizing a third algorithm, called the Representative Training Algorithm, the network is able to guide the experimental design process and to substantially reduce the number of required experiments when compared to other empirical modelling methods such as Analysis of Variance (ANOVA) and Taguchi techniques.

The development of this network has lead to the application of artificial neural system technology to numerous real problems in process modelling and signal data processing. One of these applications has been a highly complex welding process model called WELDBEAD that has been integrated into the Navy's WELDEXCELL workstation software.

1.2 THE *WELDEXCELL* SYSTEM: OVERVIEW

The joining of metals into fabricated components and structures is a difficult task. The most common method of joining metals is welding, but the welding process is complex and requires several important steps to be performed in a carefully integrated manner. The weld joint is first designed and engineered properly, then that design must be correctly communicated to the fabrication facility. The appropriate welding consumables, including filler metal and protective flux or inert gas, are chosen. Then the welding procedure is specified, including preheating schedules; welding parameters such as voltage, current and travel speed; and, postweld heat treating. Finally, the weld must be performed under highly skilled human guidance and control. A minor error in any of these steps, if undetected, can create an unsuitable welded component, which in later use may result in a catastrophic failure and perhaps loss of life.

An extremely complex and interrelated system of codes, specifications, tests, and inspections ensures that the vast majority of welds will never fail in service. A weld, which is a small bit of solidified metal, is expected to have the same (or perhaps better) properties as the base metal that it joins. The base metal may have undergone hours of careful and expensive heat treating and processing, yet the weld must be as corrosion resistant, as strong, as ductile, and as fracture resistant as the base metal. But the weld does not have the advantage of all of that processing.

Fortunately, a large number of engineers, designers, and welders work within the system of codes and specifications to ensure the high quality of welded joints, but this system is very expensive and requires the careful attention of many human

[1]Research conducted on this technique has confirmed the existence of at least three types of learning instabilities (local minimum being one of them) and the algorithm can avoid all three instabilities.

experts. As the availability of engineering talent in the United States continues to decrease over the next decade, Computer Aided Engineering (CAE) will take on added importance. Consequently, welding engineering and planning is an ideal application for artificial intelligence technology including expert systems and neural networks. However, no single expert system could be expected to perform the myriad of tasks required to make a welded joint. For example, there are over 100 welding processes ranging from simple flame heating to exotic laser welding; there are several hundred welding filler metals -- from plane carbon steel to elaborate chemical mixtures of alloying ingredients; and there are over 1000 different grades of steel classified by the American Society for Testing and Materials (ASTM) that may have to be joined. The possible combinations of welding process, filler metal, and steel base metal would number into the millions.

The solution to this problem is a system being developed for the United States Navy called WELDEXCELL. The beta test prototype of WELDEXCELL was delivered to Puget Sound Naval Ship Yard for testing during the summer of 1991. The expert systems needed for welding include materials selection, joint design, welding process and procedure selection. There is a standard CAD system interface to draw the design and communicate that design to the shop floor as well as a CAD interface to the robot path planner. The system also includes intelligent processing to control a complex automated welding system or robot with an array of sensors to guide it and to provide feedback for process control. The system actually being delivered to the Navy will be configured with at least two sensors, but the system is capable of operating in a multiple sensor environment.

The American Welding Institute (AWI), together with its partners, the Colorado School of Mines and MTS Systems, Inc., is developing this intelligent weld process planner and intelligent control system for flexible welded fabrication known as the WELDing EXpert manufacturing CELL (WELDEXCELL). This project has entailed development of a series of linked expert systems and neural networks acting as a computer aided engineering and planning assistant and software to download welding plans and procedures to a welding system and automated manipulator or robot for automatic execution. An intelligent welding control system is being completed which will interface to a robot system, a series of sensors, and to the welding equipment.

2.0 WELD PROCESS MODEL NEURAL NETWORK

2.1 THE WELDING PROCESS

Welding is a complex process that includes the welding arc plasma, base materials that are melting and solidifying, in some cases a wire that melts with droplets forming, and the associated equipment. None of these components have ever been completely successfully modeled using conventional empirical or theoretical approaches.

- The arc plasma is a super heated gas consisting primarily of ions and electrons in highly excited states. Even measuring the behavior of the plasma is difficult since the plasma is both a radiating and an absorbing semi-transparent heat source. Temperature models and measurements have shown that welding arc plasma temperatures range up to 20,000° Kelvin. A temperature distribution in the plasma or even at the surface of the base metals and the wire is, at best, an un-verified approximation.

- Both the melting and solidification of the base metals is a heat flow problem, known as the Stefan Problem, that has no known theoretical solution. Work in this area has shown reasonably good results with finite element modelling. However, the best of these codes requires several tens of hours of supercomputer time at a cost that is prohibitive for virtually all real welding applications.

- The melting of the filler wire is a somewhat less complex problem, except that when the filler wire is one of the electrodes in the welding process, the behavior is both heat flow as well as electrical characteristics and behaviors at very high current densities. For most welding, where the wire is the electrode, no models exist that provide accurate melting and droplet formation information.

- The entire process is significantly affected by the electrical characteristics of the welding power supply and associated equipment. This equipment is generally well characterized as to its static behavior, but no methods have been discovered to model the dynamic activity of these systems.

Today, the methodology applied to developing a plan or "weld schedule" for a joining operation is manual. Several test welds

are completed and the results, including photographic representations of the weld bead, are recorded in welding procedure Qualification Records (PQR). An experienced welding engineer places several of these PQR's on his desk, and makes "educated guesses" as to the effects of the various control parameters.

2.2 THE WELD NEURAL NETWORK MODEL: *WELDBEAD*

Neural networks have been developed to model the Gas Tungsten Arc Welding (GTAW) and Gas Metal Arc Welding (GMAW) processes and to provide several outputs regarding the resulting weldment. The model operates on standard PC type computers and can provide solutions at a speed of approximately 1/100 sec. Operating in the inverse mode, the model requires 1-2 seconds to determine a solution. The outputs include performance of the welding process such as arc stability and spatter, bead characteristics such as penetration, leg widths, bead undercut, appearance, and ease of slag removal. These networks use welding voltage, current, wire feed rate, and travel speed as inputs. The models are used to predict bead characteristics, given values for the inputs. In addition, the networks can be used for an inverse search to find near optimum voltage, current and travel speed values, given desired bead characteristics and weld performance. The system allows the user to vary importance factors on each of the desired welding results parameters.

These model networks are the heart of an intelligent control planning system in the WELDEXCELL software. The objective of the associated intelligent control system is not to control the primary independent welding parameters (e.g. voltage, current, travel speed, etc.) but to control the final weld quality. That is, control of weld quality parameters such as bead appearance, penetration, amount of spatter, etc. is the goal of an intelligent control system. Sensor data is not particularly useful unless it can be both analyzed and used to control the weld quality parameters. These model networks provide that capability.

One of the neural network models of the weld has inputs of voltage, current, and travel speed and outputs of resulting weld bead height, top and bottom width, and amount of penetration. Figures 1 and 2 show the user interface to the GTAW network. On the screen are several mouse sensitive slide bars that can be set by the user or observed by the user. The top row of slide bars represent the outputs of the forward network. The second row of slide bars are mouse sensitive and represent voltage, current, wire feed rate, and travel speed. In addition, the model allows for changes in the gap (opening between the base metals) and the base metal thickness. Finally, many welds are "tacked" with small bits of weld to hold the pieces in place before and during the actual weld. The model includes the expected changes to the weld when it is made over one of these "tacks". Figures 1 and 2 show examples of the GTAW butt weld neural network with macrographs of the resulting test weld experiments.

3.0 CONCLUSIONS

This program has shown that it is possible to develop models of highly complex and non-linear processes using the technology of neural networks. The model can provide all of the information necessary to produce a plan for an automated process operation. In addition, the neural network model is able to be operated in the inverse mode which provides a capability for incorporation of this technology into intelligent control systems.

4.0 ACKNOWLEDGEMENT

The authors wish to acknowledge the support of the American Welding Institute. In addition, for the WELDEXCELL application of the WELDBEAD software, resources were provided by the Navy Manufacturing Technology Program.

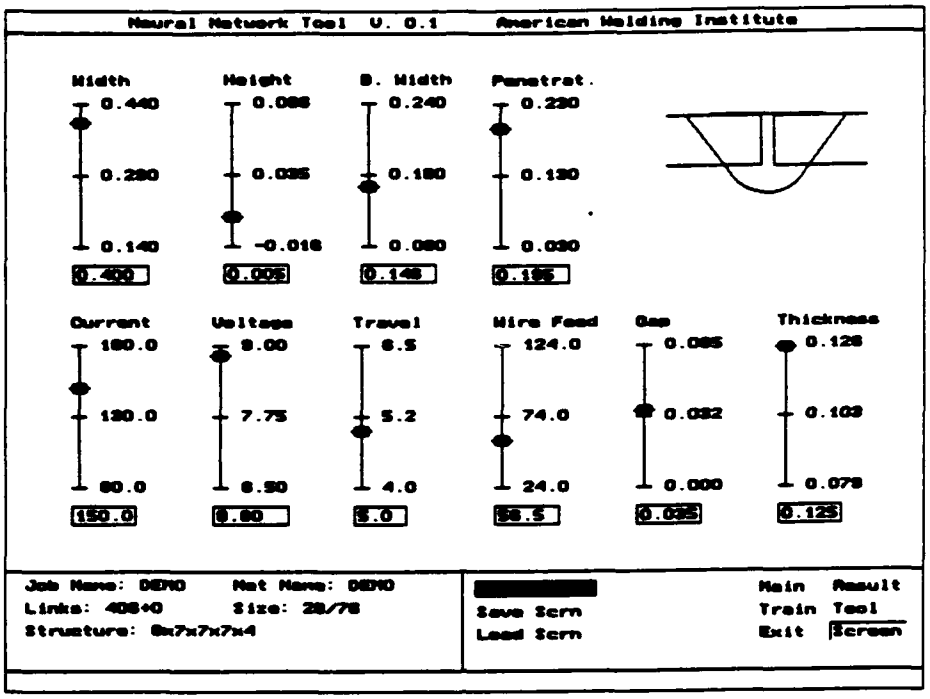


Figure 1. Neural network solution for the butt weld. Selected parameters of: voltage = 8.8 volts, current = 150.0 amperes, travel speed = 5.0 inches per minute, and wire feed rate = 56.5 inches per minute. In addition, the weld is being made in material of thickness = 0.125 inches and with a gap of 0.035 inches. The resultant shape parameters and bead graphic are shown in the interface.

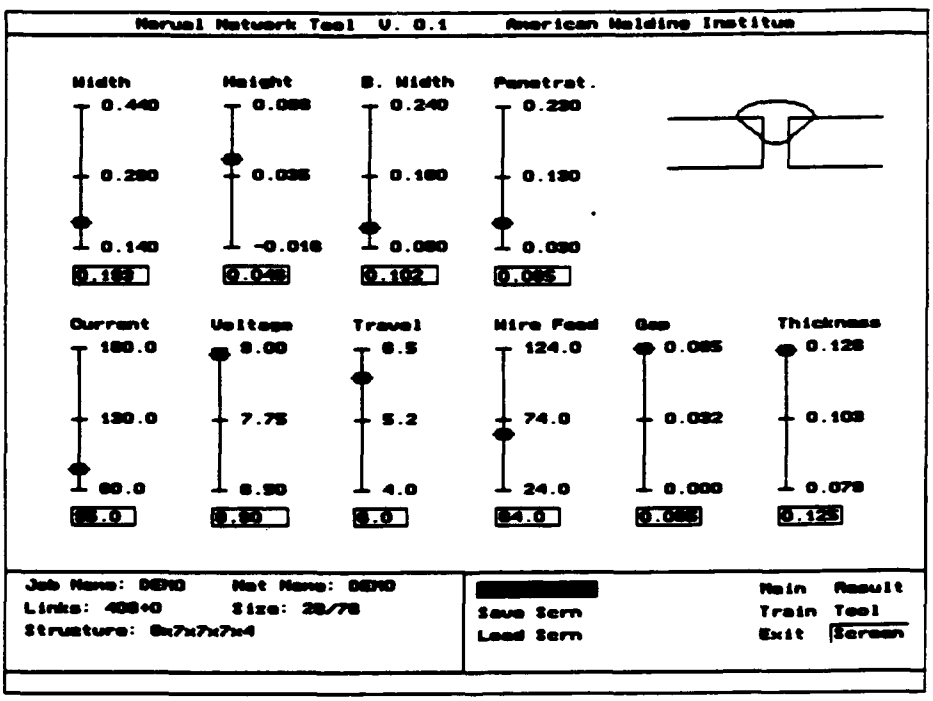


Figure 2. Neural network solution for the butt weld. Selected parameters of: voltage = 8.9 volts, current = 95.0 amperes, travel speed = 6.0 inches per minute, and wire feed rate = 64.0 inches per minute. In addition, the weld is being made in material of thickness = 0.125 inches and with a gap of 0.065 inches. The resultant shape parameters and bead graphic are shown in the interface.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop

24-26 August 1992 GACIAC PR 92-02

Using Neural Networks to Analyze Nondestructive Evaluation Patterns[1]

W. E. Bond and D. C. St. Clair[2]
McDonnell Douglas Research Laboratories, St. Louis, MO
Email: bond@mdcgwy.mdc.com

Abstract

Nondestructive evaluation (NDE) techniques represent a class of nonintrusive processes used to detect anomalies in materials without requiring the destruction or disassembly of the object being tested. Aircraft structures are an important area for the application of NDE, since both metallic and composite structures must be monitored for cracks and abnormalities. Various NDE techniques are used to verify the integrity of aircraft structural parts prior to assembly, during assembly, and during maintenance operations.

Current NDE practices are labor intensive and rely heavily on the judgment of human experts. The goal of our NDE research program is to partially automate NDE, to enhance existing techniques for detecting anomalies, and to reduce testing costs. To achieve this goal, neural network algorithms were used to build and maintain systems that can classify NDE results. Preprocessing techniques were used to identify significant "features" in the waveforms. These features were then "learned" by the neural networks and a system was built which could characterize new test results. The objective is to use the features learned to identify or classify new information.

Background

Current McDonnell Douglas Corporation (MDC) efforts have focused on automating A-scan ultrasonic techniques. A-scan was chosen after preliminary research predicted a high likelihood of success. Two other papers have discussed some of the previous efforts in this area[3] [4].

A-scan ultrasonic inspection produces a waveform showing reflected or through-transmitted ultrasonic sound amplitude as a function of time (depth) for a particular location on the test specimen. By examining the sound energy that is transmitted or reflected, it is possible to make determinations about the internal structure of the material[5] [6]. The ultrasonic waveform

---

1 Work supported by the McDonnell Douglas Independent Research and Development program.
Export Authority: 22 CFR 125, 4(b) (13).

2 D. C. St. Clair is a Professor of Computer Science at the University of Missouri-Rolla Engineering Education Center in St. Louis as well as a Visiting Principal Scientist at McDonnell Douglas Research Laboratories.

3 Amirfathi, M.M, Morris, S., O'Rorke, P., Bond, W.E., and St. Clair, D.C., *Pattern Recognition for Non-Destructive Evaluation*, **Proceedings of the 1991 IEEE Aerospace Applications Conference**, February 3-8, 1991.

4 Bond, W.E., St. Clair, D.C., Amirfathi, M.M., Merz, C.J., and Aylward, S., *Neural Network Analysis of Nondestructive Evaluation Patterns*, **Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing**, pp. 643-650.

5 Gallagher, J.P., Giessler, F. J., Berens, A. P., and Engle, R. M., **USAF Damage Tolerant Design Handbook: Guidelines for the Analysis and Design of Damage Tolerant Aircraft Structures**, Wright-Patterson Air Force Base, Air Force Report No. AFWAL-TR-82-3073, 1984.

6 McMaster, R.C., Ultrasonic Test Principles, **Nondestructive Testing Handbook**, Vol. II, Section 43, 1959.

produced at a location containing an anomaly will differ (sometimes significantly, sometimes subtly) from an "expected" waveform.

For example, if the test probe was placed at defect-free location "A" in Figure 1 the resulting waveform would look like Figure 2. Figure 3 shows the waveform that would be obtained if the test probe were placed above a defect, such as at location "B." Figure 3 illustrates a fairly large, obvious defect in a test sample which has been constructed to verify the operation of the test equipment. Not all defect indications are this pronounced, especially when the part geometry is complicated or the defect location is close to either the front or back surface.

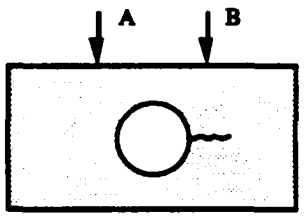


Figure 1. Example test situation

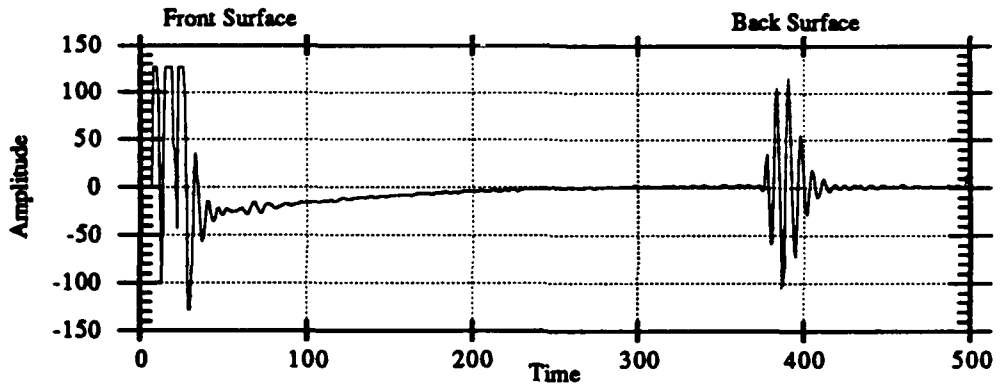


Figure 2. Typical A-scan representation of no-defect

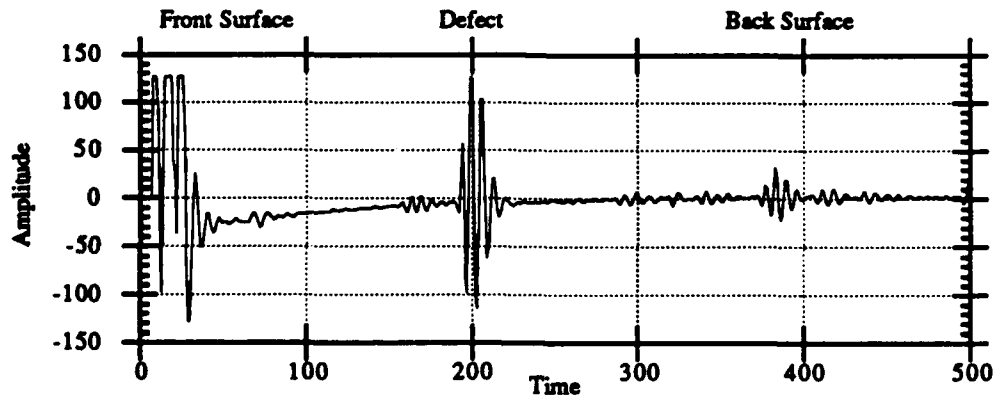


Figure 3. Typical A-scan representation of a significant defect

The "features" or "patterns" seen in a waveform are indicative of the conditions found in the test specimen. Since these features should be fairly consistent for a specific geometry/material and are directly related to specific conditions, it would seem that a system could be constructed to recognize these features. Further, NDE test procedures precisely define the

conditions under which a test should be performed. These procedures define the type of equipment, type of probe, equipment settings, and location of the probe. In addition, a calibration sample is often provided to align and verify operation of the test equipment before actual readings are made. This control is intended to minimize the effects of varying environmental conditions. This environmental control should contribute positively to the success of using neural network techniques in NDE.

To characterize the waveforms, several different types of preprocessing were investigated. From this investigation, it was clear that selecting appropriate preprocessing techniques for the data was critical to the success of the neural network algorithms. To date, time and frequency-domain preprocessing has been investigated. A time-domain (TD) technique that highlighted waveform features and minimized noise was developed. The Fast Fourier Transform (FFT) was also used to characterize waveforms. Both these techniques have produced excellent results, which are reported below. The drawback of these techniques is that training data must be available for each specific part/material/geometry. What is required is a set of parameters, independent of specific parts and geometries, which can be used to characterize test patterns. The development of an appropriate set of parameters is currently being studied.

Neural Network Algorithms

After a review of several types of learning techniques[7], two neural network algorithms were chosen to develop the NDE classification systems. The first technique used was a Backpropagation algorithm[8] and the second was an algorithm, developed at MDC, called SMART2[9]. The SMART2 algorithm is a supervised version of the Adaptive Resonance Theory algorithm, ART2, described by Carpenter and Grossberg[10]. SMART2 trains several times faster than Backpropagation and produces accuracy very similar to Backpropagation.

Results

To evaluate these preprocessing techniques and neural network algorithms, a set of experiments was designed which evaluated test results from three different parts. Multiple readings were taken using hand-held test equipment on the first two parts. Data was collected on the third part by test equipment which automatically positions the probe and records data.

The first group of data was obtained from a solid laminate standard-composite panel at the part locations shown in Figure 4. Teflon inserts were placed at different locations and at different depths during the fabrication of the panel to simulate defects in a composite material. Teflon inserts located along the edges of the sample were removed after the sample cured. This represented delaminations at those locations. The panel is shown schematically in Figure 4. The part was tested at each of the numbered test points as indicated by the arrows shown in the cross section view. Positions 1, 5, 6, 10, 11, and 15 are delaminations while positions 2, 3, 4, 7, 8, 9, 12, 13, and 14 are inserts of different sizes which represent inclusions. A series of readings were also taken at other locations to obtain readings from "good" portions of the panel. These samples are denoted by position 16.

The data obtained from the 16 different locations was preprocessed. Then, the data was grouped into 4 and 7 classes. Four classes were formed by grouping test points that correspond to a particular depth in the composite panel. Class 1 corresponds to locations 1, 2, 3, 4, and 5. Class 2 corresponds to locations 6, 7, 8, 9, and 10. Class 3 corresponds to locations 11, 12, 13, 14, and 15. The fourth class corresponds to location 16. Seven classes were formed by grouping together test points

---

7 St. Clair, D.C., Bond, W.E., Rigler, A.K., and Aylward, S., *An Evaluation of Learning Performance in Backpropagation Neural Networks and Decision-Tree Classifier Systems*, **Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing**, Vol. II, pp. 643-650.

8 Rumelhart, D. E., Hinton, G. E., and Williams, R. J., *Learning Interior Representation by Error Propagation*, **Parallel Distributed Processing**, Vol. 1, Ch. 8, D.E. Rumelhart and J.L. McClelland eds., MIT Press Cambridge, Mass. 1986.

9 Merz, C.J., St. Clair, D.C., and Bond, W.E., *SeMi-supervised Adaptive Resonance Theory: SMART2*, **Proceedings of the 1992 IJCNN International Joint Conference on Neural Networks**, Vol. III, pp. 851-856.

10 Carpenter, G.A., and Grossberg, S., *ART 2: self-organization of stable category recognition codes for analog input patterns*, **Applied Optics**, Dec. 1, 1987, No. 23, pp. 4919-4930.

that correspond to a particular type and depth of defect: Class 1 - locations 2, 3, 4; Class 2 - locations 1, 5; Class 3 - locations 7, 8, 9; Class 4 - locations 6, 10; Class 5 - locations 12, 13, 14; Class 6 - locations 11, 15 and Class 7 corresponds to location 16. To produce training and testing data for the algorithms, each dataset was randomly split into two datasets, with 70% of the data being used for training and 30% for testing.

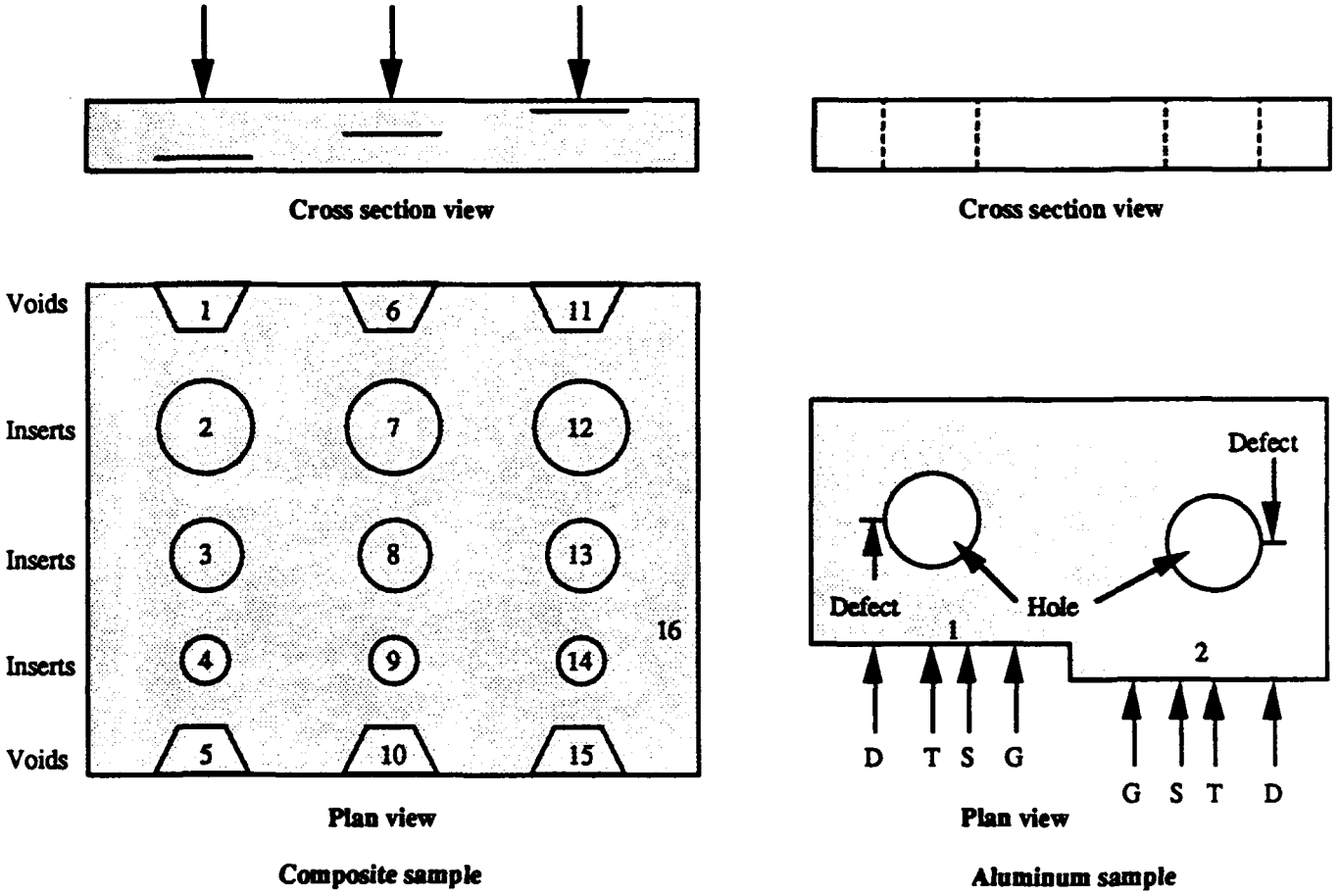


Figure 4. Schematic of the parts

The percent accuracies shown in Table I indicate each network's ability to associate test instances with their correct classification. The results obtained were excellent with both algorithms. Inclusions and delaminations clearly produce distinct waveforms which can be discerned by the systems. The seven classes provide a much better distinction among defects than do the four classes.

	SMART2 Test Accuracy		BP Test Accuracy	
	TD preprocessing	FFT preprocessing	TD preprocessing	FFT preprocessing
Composite Panel				
4	99.0		99.8	
7	98.3	100.0	99.1	100.0
Calibration Fixture				
1	100.0	99.0	100.0	99.0
2	100.0		100.0	

Table I. Results

To evaluate the performance of these techniques on an aluminum sample, a set of test readings was obtained from a calibration fixture that corresponds to an aluminum aircraft part (see Figure 4). Two holes are located in the calibration fixture. The "defects" in the fixture were introduced by machining small slots. Readings were taken at two different locations on the fixture (denoted 1, 2). At each location, four different classes of data were obtained corresponding to the defect (D), the top of the hole (T), the side of the hole (S), and a portion of the sample clear of both the hole and the defect (G). Access to inspect the actual aircraft part is limited to a single side. In addition, it is difficult to determine exactly where the holes are located without careful measurement.

The training and test data was obtained from the same calibration fixture but the readings were taken at a different time. Training and test data was preprocessed. Excellent results were obtained, as shown in Table I. The two neural network techniques provide essentially the same results.

The third part which was evaluated is an actual aircraft part. This is a complex part which is fabricated by bonding two metal skins to a closure. The space between the skins is filled with a metal honeycomb. This part is difficult to inspect because of the multiple materials, the bond, and changing geometry. The task was to determine areas where the bond between the metal parts was not fully developed so that repairs could be effected. The part used for testing had been carefully inspected by hand, revealing several areas where the bond was not fully developed.

Data was collected from the part using automated test equipment. A "slice" of data was selected which contained one of the undeveloped bond areas plus multiple examples of a good bond; this data was used to train the system. Data from the entire part was analyzed using the resulting system. The results were good. Each undeveloped bond area that had been identified by hand inspection was also identified by the system. The system also identified several other areas as having an undeveloped bond. These results are still being studied. The system significantly reduced the amount of data requiring human interpretation. The data reduction was from 30 to 110 times depending on the type of preprocessing used. The results indicate that these techniques were successful even for this complicated part.

Summary

This project has demonstrated that, for these samples, an automated system can be constructed to classify NDE test results. An automated system could significantly decrease testing time and provide consistent test results. Work continues on improving the preprocessing and neural network techniques.

Acknowledgments

This work was initiated in 1990 as a cooperative effort between the Artificial Intelligence groups at Douglas Aircraft Company (M. M. Amirfathi) and McDonnell Douglas Research Laboratories and the Nondestructive Evaluation Group at Douglas Aircraft Company (M. R. Collingwood and D. J. Hagemaier). During 1991, the McDonnell Aircraft Company Neural Network Lab (S. Aylward) became part of the effort and test data and domain expertise was supplied by McDonnell Aircraft Company Nondestructive evaluation groups (K. Jepson, H. Parzuchowski, M. Reighard, and D. Theilen). This work also benefitted from two university contracts, one with the University of Missouri - Rolla, Engineering Education Center in St. Louis and the other with the University of California, Irvine.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

An Application of Neural Network Techniques to Statistical Process Control Chart Interpretation

Mark Shewhart

Air Force Materiel Command (AFMC)
Center for Supportability and Technology Insertion (CSTI)
Process Improvement Division (CSTI/PIAP)
Wright Patterson AFB, Ohio 45433

Abstract.

The Special Projects Office of the Center for Supportability and Technology Insertion (CSTI) has developed a hybrid neural-network/expert-system software tool (*SPC Version 1.1*) which automates the process of constructing and interpreting control charts. *SPC Version 1.1* is a highly successful example of the practical application of neural network and expert system techniques. The goal of this paper is to highlight the implementation and application of a software system which uses neural network techniques to solve real people's problems. *SPC Version 1.1* is **FREE** to all government agencies.

INTRODUCTION

The Air Force Materiel Command (AFMC) has provided TQM and Quality Control training to its employees for several years now. In particular, Statistical Process Control has been emphasized in this effort. While many data collection and process improvement efforts have been undertaken within AFMC, the SPC Quality Control tools have been under-utilized due to the complexity of manual application and interpretation. In response, CSTI has developed a software tool which automates the graphical and statistical techniques used in Statistical Quality Control. The most current version of the software, *SPC Version 1.1*, automates eight types of control charts and seven types of graphical tools: Run Chart, Moving Range Chart, XBar-R Chart, XBar-S Chart, PN Chart, P Chart, C Chart, U Chart, Bar Graph, Pareto Diagram, Pie Chart, Histogram, Frequency Polygon, Ogive (CDF), and Scatter Diagram.

CONTROL CHARTS

A run chart is a plot of a process measurement (e.g. bore diameter or time to process an insurance claim) on the vertical axis (y-axis) against time on the horizontal axis (x-axis). A control chart is simply a run chart with statistically determined upper (Upper Control Limit - UCL) and lower (Lower Control Limit - LCL) lines drawn on either side of the process average. These limits are calculated by running a process untouched, taking samples of the process measurement, and applying the appropriate statistical formulas (references [3-9]). An example of a control chart is given below in **FIGURE 1**.

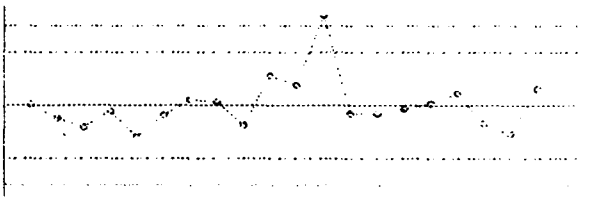


Figure 1 Sample Control Chart

The random fluctuation of points within the limits results from variation built into the process. Such random variation is natural, results from common causes within the system (e.g. design, choice of machine, preventative maintenance, etc.), and can only be affected by changing the system itself. However, points which fall outside of the control limits or which form "unnatural" patterns indicate that some of the variation within the process may be due to assignable causes. Assignable causes of variation (e.g. measurement errors, unplanned events, freak occurrences, etc.) can be identified and result from occurrences that are not part of the process.

The purpose of drawing the control chart is to detect any unusual causes of variation in the process, signalled by abnormal points or patterns on the graph. The CSTI developed software tool automatically identifies nine types of patterns which suggest the presence of assignable causes of variation in a process. An example of one of these patterns is given in **FIGURE 2.** Each of the nine patterns is associated with generic advice about what may be happening at that point in the process.

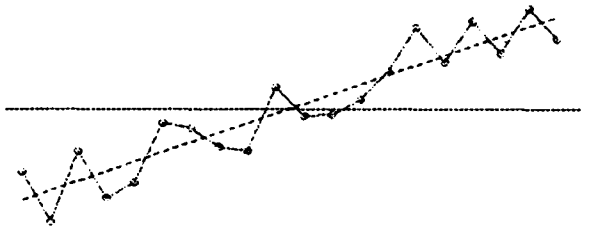


Figure 2 An Increasing Trend
Example of a Pattern Which Suggests the Presence of Assignable Causes of Variation

SOFTWARE FUNCTIONALITY

(1) ***SPC*** determines which type of control chart is appropriate by asking a series of questions about the nature of the user's process data.

(2) ***SPC*** graphically displays the chart(s) selected in (1).

(3) ***SPC*** identifies 9 pattern types in the chart(s) which suggest the presence of assignable causes of variation : increasing trends, decreasing trends, shifts up, shifts down, cycles, runs, stratification, freak patterns, and freak points.

(4) ***SPC*** graphically displays and highlights each chart pattern identified in (3).

(5) ***SPC*** displays text in a hypertext fashion which provides generic advice on the meaning of each chart pattern identified in (3).

SPC 1.1 CUSTOMERS

SPC is a generic tool whose use is not limited to a particular area of application, Air Force command, or federal agency. ***SPC Version 1.1*** is currently being used by academia, industry, and the federal government. Although SPC is only in its second release, it has developed a wide-ranging customer base which is growing at an exponential rate. SPC is comparable or superior to commercial packages, more user-friendly, less expensive (free), and CSTI offers its customers the ability to directly influence future enhancements to the software. Over 2500 copies have been directly distributed to working level personnel in the federal government.

SOFTWARE DESIGN

The basic approach to developing ***SPC*** was to integrate machine learning, expert systems, hypertext, and conventional programming techniques. The machine learning portion of ***SPC*** was developed using the Abductory Induction Mechanism (AIM) by AbTECH Inc. The expert system portion of ***SPC*** was prototyped using an embedded application of the forward chaining expert system tool CLIPS along with a generic end-user interface also developed by CSTI. Due to the size of CLIPS and the limitations of MS DOS, the knowledge bases used for this application were encoded directed into C++ code. The hypertext portion of ***SPC*** was developed using a hypertext authoring and display tool called ***HYPERSHOW*** developed by the Special Projects Office of CSTI. Turbo C++ was used as the conventional language into which the machine learning, expert system, and hypertext applications were embedded.

The task for the machine learning portion of ***SPC*** is to classify every sub-sequence of the control chart according to the presence or absence of five specific chart patterns : increasing trends, decreasing trends, shifts up, shifts down, and cycles. The remaining four chart patterns are identified by conventional methods.

The expert system is initially utilized to help the user select the appropriate type of control chart. This determination is based upon the type of data being collected, the number of data points, and the constancy of the sample sizes.

Another function of the expert system is to interpret the classification results of the trained AIM Network. A control chart with 40 data points will generate over 600 classification results; with nine types of patterns this amounts to over 5500 individual pieces of classification information. This interpretation function represents an ideal expert system application. What requires a few hundred lines of difficult-to-comprehend C code can be implemented using an expert system with only three simple rules! This classification information is then boiled down to about one to ten patterns which are reported to the final expert system application.

The final role of the expert system is to provide advice via the hypertext facility based upon the types of charts and the chart patterns present. The advice currently provided by ***SPC*** is of a generic nature. For example,

"A shift up in the R chart indicates that the process is becoming less consistent. This may be due to some sudden change in the process."

However, the knowledge base is designed to allow for quick modifications to provide process specific advice. For example,

"A shift up in the R chart has historically been associated (90%) with a loose bearing in the preprocessing machine."

Conventional software is used to graphically display the control charts, utilize the AIM Networks, provide an end-user interface, and integrate the entire application.

Role Of Machine Learning in Control Chart Interpretation

The task of chart interpretation can be summarized as follows. A control chart is simply a sequence or array of floating point numbers. The art of chart interpretation is to determine whether or not sub-sequences similar to several standard patterns are present within the chart. These patterns include trends, shifts, and cycles.

The function of the machine learning tool is to generate code (trained AIM Networks) which can effectively classify a specific sub-sequence of a control chart (array) according to the presence or absence of several standard patterns. With this classification function generated by machine learning techniques, all sub-sequences of the control chart are exhaustively (conventionally) classified by five AIM Networks. The AIM Network classification results are asserted into the fact-list of the CLIPS expert system application.

Justification For The Use Of Machine Learning Techniques

Machine learning techniques are used to classify five types of chart patterns - increasing trends, decreasing trends, shifts up, shifts down, and cycles. We could find no references which provide an algorithm for determining whether or not a sequence of real numbers is representative of one of these patterns. In fact, most references on control charts define these patterns by example! The most mathematical approaches to this problem are found in references [1,2] on time series analysis and forecasting. Despite being mathematical in nature, these references still do not describe a deterministic decision procedure. Rather, they provide mathematical heuristics. Sample rules-of-thumb for a times series of length N are given below:

(1) The number of increasing steps in an increasing trend may be significantly larger than (N-1)/2.

(2) The number of discordances in a decreasing trend is usually larger than the expected number of discordances in a random sequence which equals N*(N-1)/4.

(3) The autocorrelation coefficient sequence of a cycle is usually cyclic.

(4) The average of the first half of a shift down is always greater than the average of the second half.

Notice that most of these heuristics are in the form of rules with confidence factors. This would seem to suggest the possibility of using a production system for the classification procedure. However, it is almost always the case that the pattern-type (the attribute for which we wish to determine a value) is on the left-hand side of the rule.

This is very similar to some medical diagnosis problems whose domain knowledge is in the form *"Disorder A usually causes symptoms 1, 3, & 4 and may cause symptom 2."* In cases such as these, the best knowledge-based approach is to use some form of a Hypothesize-and-Test (HT) model. Although the HT approach appears to model the domain very well, we did not pursue this option since we do not have access to a HT development tool which can be easily embedded into other applications.

Attempting to implement such applications using a rule-based system with confidence factors ultimately boils down to an iterative process of re-adjusting confidence factors and re-testing the rule base on a set of examples. This iterative process, however, is quite analogous to the process of training a neural network or a machine learning tool on a set of examples. Given this analysis and the fact that most references on control charts define these patterns by example, we elected to implement a portion of the classification process using a machine learning tool.

Representation Of Control Chart Sub-sequence

The function of the machine learning tool is to classify a specific sub-sequence of a control chart according to the presence or absence of several standard patterns. A key question relating to the use of machine learning tools, is how do we represent an arbitrary length sub-sequence of an arbitrary length sequence of numbers as a fixed length vector of real numbers. The approach is to represent a sub-sequence of a control chart as a fixed length vector of

statistical features.

Twenty (20) statistical features are extracted from each sub-sequence X[1..N] under consideration. Features 1 - 10 are raw statistical features such as linear regression coefficients, the standard deviations of the first and second halves of the sequence, and the Box-Pierce Q-statistic [1, p 269]. Features 11 - 20 are Boolean type indicator variables such as whether the sequence has ten or more points and whether one of the regression coefficients is "low" in magnitude. Most of these Boolean indicators were included as a result of a database analysis on the training sets which produced "statistics" such as *"90% of increasing trends have a linear regression coefficient greater than 0.8"*. A more detailed explanation of the 20 statistical features used to represent a chart subsequence to the AIM Networks is given in **APPENDIX A.**

Training And Test Sets For Machine Learning Tool

Over 70,000 sample chart sub-sequences were generated to train and test the AIM Networks. Most of these sub-sequences were generated by adding random noise of various magnitudes to existing control charts with existing patterns. Each chart sub-sequence generated a training/test vector of dimension 25 - 20 real-valued Network inputs (statistical features) and 5 bi-polar (-1 or 1) outputs. One AIM Network was trained for each of the 5 outputs. Each AIM Network required from two to six hours to train on a 386 machine with math co-processor.

Machine Learning Test Results

The results of the AIM Networks applied to control chart patterns not present in the training sets were quite good. The increasing trend network classified the test data 97.5% correctly, the decreasing trend network 97.3%, the shift up network 98.8%, the shift down network 98.8%, and the cyclic network 92.0%. Most (99%) of the errors in the cyclic network occurred in the short saw-toothed patterns with added noise. If the saw-toothed pattern is deemed by experts/customers to be very important, a separate network could be developed for the saw-toothed pattern. This would increase the overall cyclic percent correct to about 96% and provide a better recognition rate for noisy, short saw-toothed patterns.

CONCLUSION

SPC Version 1.1 was released in March 1992 and *SPC Version 1.2* is scheduled for release in September 1992. CSTI plans to provide software enhancements to *SPC* based upon continuous customer feedback and demand. *SPC* has a wide customer base which is continuing to grow as word of this powerful, easy-to-use software tool spreads. **Government agencies now have a no-cost option when looking for Statistical Quality Control and Process Improvement software.** In the words of the Air Force Logistics Command's Quality Programs Directorate (AFLC/QP), *"CSTI has applied a technology, artificial intelligence (neural nets), and developed a system that can be used by real people to solve real problems"*. Reprints of this paper and copies of *SPC Version 1.1* are available to government agencies and domestic US companies.

ABOUT THE AUTHOR

Mark Shewhart is an Artificial Intelligence Systems Engineer for the Process Improvement Division of the Center for Supportability and Technology Insertion (CSTI) at Wright-Patterson AFB, Ohio. Mr Shewhart has a B.S. in Computer Engineering and an M.S. in Applied Mathematics from the University of Illinois at Urbana and an M.S. in Computer Science from Wright State University in Dayton Ohio. Mark's primary duties involve the management and development of CSTI's SPC software tool. Interestingly, Mark Shewhart's grandfather and Walter Shewhart, the "Father of Statistical Process Control", were first cousins.

REFERENCES

[1] Spyros Makridakis and Stephen C. Wheelwright, "Forecasting: Methods and Applications",Wiley/Hamilton, 1978.

[2] Sir Maurice Kendall and J Keith Ord, "Time Series", Oxford University Press, 1990.

[3] SPC Course Materials, Decision Dynamics Inc., 1990

[4] Kaoru Ishikawa, "Guide to Quality Control", Asian Productivity Organization, 1982.

[5] Perry Johnson Inc., "SPC Chart Interpretation", Perry Johnson, Inc., 1987.

[6] J.M. Juran, Dr. Frank M. Gryna, Jr., and R.S. Bingham, Jr., "Quality Control Handbook",Third Edition, McGraw-Hill, 1974.

[7] Western Electric Company, "Statistical Quality Control Handbook", Western Ellectric Co., Inc., 1958.

[8] H. Besterfield, "Quality Control", Second Edition, Prentice-Hall.

[9] Douglas C. Montogomery, "Introduction to Statistical Quality Control".

APPENDIX A

Statistical Features Used To Represent Chart Subsequences

(1) RMS_SU - This is the root-mean-squared difference between X[1..N] and an "ideal" shift-up pattern.

(2) RMS_SD - This is the root-mean-squared difference between X[1..N] and an "ideal" shift-down pattern.

(3) A - This is the simple linear regression coefficient when trying to approximate the time series X[t] using X[t] = A + Bt.

(4) B - This is the simple linear regression coefficient when trying to approximate the time series X[t] using X[t] = A + Bt.

(5) SIGMA_1 - This is the standard deviation of the first half X[1..N/2] of the sequence X[1..N].

(6) SIGMA_2 - This is the standard deviation of the second half X[N/2+1..N] of the sequence X[1..N].

(7) R_root_N_r - The percentage of the first N/4+1 autocorrelation coefficients r(k) for which abs(r(k)) > 1.96/sqrt(N).

(8) CHI_SQ_TEST - This is the Box-Pierce Q-statistic which is capable of determining whether several autocorrelation coefficients are significantly different from zero. Defined in reference [1,p 269]

(9) CONCORD - This is the number of concordances Q in X[1..N] divided by the maximum possible number N(N-1)/2 of concordances. Defined in reference [2,pp 21-23].

(10) DISCORD - This is the number of discordances P in X[1..N] divided by the maximum possible number N(N-1)/2 of discordances. Defined in reference [2,pp 21-23].

(11) TEN_PLUS - An indicator variable used to indicate if X[1..N] has length less than ten. This is important since many statistical significance tests are ineffective for small sample sizes.

(12) CCRD_LOW - An indicator variable used to indicate whether CONCORD is less than 0.7. The value of 0.7 was chosen since a database analysis indicated that a high percentage of increasing trends had CONCORD > 0.7.

(13) DCRD_LOW - An indicator variable used to indicate whether DISCORD is less than 0.7. The value of 0.7 was chosen since a database analysis indicated that a high percentage of decreasing trends had DISCORD > 0.7.

(14) HIGH_ISD - An indicator variable used to indicate whether RMS_SD is greater than 1.8. The value of 1.8 was chosen since a database analysis indicated that a high percentage of shifts-up had RMS_SD > 1.8.

(15) HIGH_ISU - An indicator variable used to indicate whether RMS_SU is greater than 1.8. The value of 1.8 was chosen since a database analysis indicated that a high percentage of shifts-down had RMS_SU > 1.8.

(16) GOOD_INC_MM - An indicator variable used to indicate when the sequence minimum was early and the sequence maximum was late. The first 20% and last 20% was chosen since a database analysis indicated that a high percentage of increasing trends had their minimum and maximum within the first 20% and last 20% respectively of the sequence.

(17) GOOD_DEC_MM - An indicator variable used to indicate when the sequence maximum was early and the sequence minimum was late. The first 20% and last 20% was chosen since a database analysis indicated that a high percentage of decreasing trends had their maximum and minimum within the first 20% and last 20% respectively of the sequence.

(18) HIGH_R_root_N - An indicator variable used to indicate whether R_root_N_r is greater than 0.1. The object of introducing this variable was to help draw a distinction between random sequences and cycles. The value of 0.1 was chosen since a database analysis indicated that a high percentage of cycles and a low percentage of random sequences had R_root_N_r > 0.1.

(19) SMALL_A - An indicator variable used to indicate whether the absolute value of A is less than 0.8. The object of introducing this variable was to help draw a distinction between random sequences or cycles and the other chart patterns. The value of 0.8 was chosen since a database analysis indicated that a high percentage of cycles and random sequences and a low percentage of other types of patterns had abs(A) < 0.8.

(20) MAYBE_CYCLE - An indicator variable used to indicate when both R_root_N_r > 0.1 and ABS(A) < 0.8. This is the logical AND of variables 18 and 19.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

Cell Management Using Neural Network Approaches

DerShung Yang
Beckman Institute
University of Illinois
Urbana, IL 61801
yang@cs.uiuc.edu

James H. Garrett, Jr.
Dept. of Civil Eng.
Carnegie Mellon University
Pittsburgh, PA 15213
garrett@ce.cmu.edu

Doris S. Shaw
Construction Eng. Res. Lab.
U. S. Army Corps of Engineers
Champaign, IL 61826
shaw@bambi.cecer.army.mil

Abstract

Computer-Aided Design (CAD) systems have become an industrial standard for producing high-quality electronic drawings. Due to the frequent uses of replicated patterns, most CAD systems provide commands to facilitate grouping of primitive drawing entities as complex entities, such as *cells*. Each cell can be manipulated as a whole. Typical examples of cells for architecture design are doors, windows, etc. Cells improve design quality and speedup the design process. Cells are especially important in a cooperative environment where different persons use different software packages to access the same piece of information by referring to the unique representation scheme enforced by cells. Nevertheless, because of the enormous number of cells, cells are not easy to manage.

This paper presents a prototype system for managing cells using neural network technology. Our system is embedded in a CAD system and intends to help the CAD user utilize standard cells effectively. Our prototype system intends to lessen the management difficulty by automatically identifying the cells that are similar to what the user is drawing and suggesting that the user replace the drawing with standard cells. This process is done while the user is drawing objects on the screen.

Transformation-invariant pattern recognition is the most important research issue for this system. We have studied and compared four different approaches appearing in recent neural network literature. We chose to employ Le Cun, *et al*'s Zipcode Net preceded by Yüceer and Oflazer's ST blocks and used several rotated images as training examples to solve the rotation problem.

1 Introduction

Computer-Aided Design (CAD) systems have become an industrial standard for producing high-quality electronic drawings. Due to the frequent uses of replicated patterns, most CAD systems provide commands to facilitate grouping of primitive drawing entities, such as lines and circles. Once a set of drawing entities have been grouped together to form a *cell*, they can be manipulated as a whole. Cells can be nested, i.e., a cell may contain another cell, and thus can be fairly complicated. Typical examples of cells for architecture design are doors, windows, pieces of furniture, etc. Cells improve design quality by forcing objects of the same type to be drawn in the same way. Certain attributes, such as dimensions, layers, and textural information, are usually attached to the cells. By copying cells, these attributes are inherited as defaults. Consistency among objects of the same type is thus preserved. Cells also speedup the design process because replicated objects do not need to be re-drawn.

In addition to improving design quality and speeding up the design process, cells can also facilitate effective communication in a cooperative environment where different persons use different software packages to access the same piece of information. By using standard cells, electronic drawings can be more easily and more accurately transferred across different platforms. Accordingly, the U.S. Army Corps of Engineers has spent significant effort on compiling a set of standard cells to be used in producing electronic drawings.

This paper first discusses the difficulties in managing cell libraries in section 2. One of the essential research issues in cell management is transformation-invariant pattern recognition. In section 3, we study and compare four different neural network approaches to transformation-invariant recognition. In section 4, we give an overview of our prototype system that utilizes various techniques from these different approaches. Finally, section 5 summarizes this paper.

2 Difficulties in Cell Management

Cell libraries are not easy to manage and use. The *Standards Manual for U.S. Army Corps of Engineers CAD Systems* consists of more than one thousand standard cells that cover seven categories. These cells are currently stored in sixteen cell libraries. When a user needs to access a particular cell, he or she needs to first attach one of these sixteen cell libraries to the current drawing and then select the particular cell out of nearly 100 cells in the attached cell library. Most of these cells are named either numerically, e.g. `MP0028` for "gas piping", or abbreviated, e.g. `SHNG` for "single hinged door". Without an enormous amount of experience with these cells, a user is unlikely to remember what cells are in the cell libraries and how to search for them. Moreover, this standard set of cell libraries is still in its initial phase of development. More cells are expected to be added and radical re-organization of cell libraries might take place, which further increases the difficulty in managing these standard cells.

One traditional solution is to provide an efficient query facility to search for cells. Since each cell usually contains a corresponding textual description, in previous work we trained a neural network to learn the associations between cells and cell descriptions. This network consists of two components: The first component is a backpropagation network in a NetTalk-like[1] setup, to detect keywords in user's queries. This approach has the advantage over traditional structural approaches (e.g., syntactic parsing) that grammatical and typographical errors can be tolerated. The second component is a simple constraint satisfaction model[2] with weights set manually. This component uses the keywords identified in the previous phase to find the cell that fits best to these keywords. In this way, the user does not have to search each cell library sequentially and can type the queries free from any syntactical restrictions. However, the disadvantage of this solution is that the user must know to ask in the first place, which is not usually the case.

Because of the vast number of cells in cell libraries, a CAD user might be unaware of the existence of certain cell objects. This problem does not pertain to novice users only. Users experienced in using cells might overlook newly added cells within the cell library, or be lost due to the re-organization of cells. Therefore, a more effective solution is to develop an intelligent cell usage assistant that volunteers plausible cells while observing what the user is drawing; this solution is the approach taken for this research. With the help of this assistant, the CAD user will be notified if a portion of his or her drawing bears strong similarity to some stored cell object. Since this assistant is interactive, omissions of the standard cell objects can be corrected in the design process as early as possible.

3 Transformation Invariant Pattern Recognition

One technical difficulty encountered in cell recognition is to recognize transformed or deformed patterns, which is inevitable when the user is drawing a cell. Martin and Pittman[3] suggested using more training examples to achieve the desired generalizability. However, due to the large number of cells, obtaining a large training set is itself problematic. Therefore, we are investigating neural network models that specialize in performing transformation-invariant recognition. The following thus describes and contrasts several such neural network models.

3.1 Four Different Approaches

Le Cun, *et al.*[4], propose a constrained backpropagation network to recognize handwritten zip code digits. The Zipcode Net utilizes many techniques from the Neocognitron[5] such as weight sharing and local connectivity. The hidden layers are composed of several feature maps. Local connectivity means that each unit in a feature map receives inputs from a small region in the preceding layer. Local connectivity greatly reduces the complexity of the network and is supported

[1] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[2] D. E. Rumelhart and J. L. McClelland. *Explorations in Parallel Distributed Processing: a handbook of models, programs, and exercises.* MIT Press, Cambridge, MA, 1988.

[3] G. L. Martin and J. A. Pittman. Recognizing hand-printed letters and digits using backpropagation learning. *Neural Computation*, 3:258–267, 1991.

[4] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[5] K. Fukushima, S. Miyake, and T. Ito. Neocognitron: a neural network model for a mechanism of visual pattern recognition. *IEEE Transactions on System, Man, and Cybernetics*, 13:826–834, 1983.

by biological evidence[6]. Weight sharing means that all the units in the same feature map perform the same function by sharing incoming weights. Since different units in a feature map receive inputs from different regions in the preceding layer, weight sharing enables a feature map to detect a local feature regardless of where it appears.

Reid, *et al.*[7], propose a third-order neural network which takes the transformation invariancy of a triangle's included angles into account. Their network is essentially a three-layer network, which differs from other three-layer networks in that the hidden nodes perform multiplications over three input units (cf. sigma-pi units[8]). By sharing weights among hidden units that accept inputs from similar triangles, it can be proved that this network can perform transformation-invariant recognition[9]. However, combinatorial explosion is inevitable if all possible triangles in the input area are to be maintained. Spirkovska and Reid[10] have suggested strategies for selectively maintaining the triangles. Our experiments used a rather small number of triangles such that the number of free parameters used in the network is comparable with that used in the Zipcode Net. Otherwise, the overhead for maintaining an excessive number of triangles makes the network of no practical interest.

Yüceer and Oflazer[11] propose the use of RST-blocks to convert the raw image to a normalized image. The origin of the input area is assumed to be in the center. The T-block first aligns the center of mass to the origin point, thereby preserving the translation invariance. The S-block then normalizes the average radius of the image to a pre-determined value and thus maintains the scaling invariance. Finally, the R-block rotates the image such that the x-axis collides with the axis of maximal variance, calculated by the Karhunen-Loéve expansion[12]. The final image is then fed into a feed-forward neural network for recognition. It is obvious that the RST-blocks approach is a general pre-processing technique and can be with any network.

Normal backpropagation is also included for comparison purpose. The next section describes our experiments and reports our results.

3.2 Comparison of Different Approaches

We tested these different neural networks using the same data and procedure. The target concept was ten 10×10 numeric digits. The input area was 16×16 pixels. The learning process was considered to be completed when all training samples were "recognized" correctly. A training sample was recognized when the target output unit had the largest activation among all the output units and was above a threshold (0.6), and the difference between the largest and second largest output activations was above another threshold (0.4). A testing sample was recognized correctly if the target output unit had the largest activation. The testing samples contained various transformed images and had no overlap with the training samples. Six different configurations for each neural network were tried and their predictive accuracies were averaged and reported in Table 1. More detailed description of this comparison will appear elsewhere.

Our results indicated that the Zipcode Net performed best compared to the Third-order Net and the Backpropagation Net. Although the Third-order Net is theoretically sound, selecting a small number of triangles to maintain in order to achieve the desired generalizability seemed non-trivial. The Zipcode Net's performance, on the other hand, seemed less dependent on the network configuration. In a noisy environment, the Zipcode Net without RST-blocks performed surprisingly well.

The RST-blocks were in general effective for removing the effect of transformations in the raw images. The networks preceded by the RST-blocks usually gained significant accuracy improvement over their counterparts that without the RST preprocessing. However, the RST-blocks quickly deteriorated when noise was present. We believed that the major reason for the RST-blocks' poor performance in a noisy environment was due to the R block. The noise was inserted randomly and uniformly into the input area. These noisy data should have less impact on the T block and the S block because of their averaging functions. However, these noisy data smoothed out differences of variances in

[6] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[7] M. B. Reid, L. Spirkovska, and E. Ochoa. Simultaneous position, scale, and rotation invariant pattern classification using third-order neural networks. *Neural Networks*, 1(3):154–159, 1989.

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[9] C. L. Giles and T. Maxwell. Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, 26(23):4972–4978, 1987.

[10] L. Spirkovska and M. B. Reid. Connectivity strategies for higher-order neural networks applied to pattern recognition. In *Proceedings of the Second International Joint Conference on Neural Networks*, volume 1, pages 21–26, 1990.

[11] C. Yüceer and K. Oflazer. A rotation, scaling, and translation invariant pattern classification system. In *Proceedings of the Sixth International Symposium on Computer and Information Science*, pages 859–869, 1991.

[12] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice Hall, Englewood Cliffs, New Jersey, 1982.

	t/d	Zipcode		Third-order		Backprop	
		w/o RST	w/ RST	w/o RST	w/ RST	w/o RST	w/ RST
Translations	1	13.72	53.92	13.54	52.74	10.66	47.57
	5	19.96	98.83	15.64	99.05	13.64	98.93
Rotations	1	14.81	26.05	17.29	17.91	10.24	16.72
	4	46.88	48.85	34.88	41.88	34.80	41.57
Scalings	1	19.83	24.83	16.00	24.00	14.00	19.33
	3	64.38	55.00	30.75	46.67	43.75	48.96
Noises	1	77.14	14.05	55.24	14.05	54.05	12.14
	2	81.67	12.50	55.00	14.72	55.56	11.11

Table 1: Predictive accuracy (in percentage) of different transformation-invariant pattern recognition approaches (t/d signifies the number of training samples per digit).

different directions and made the R block unable to correctly determine the rotation angle[13].

Using more training samples per digit also yielded the improvement on accuracy. This is a traditional way to improve the accuracy. We believed that using more training samples was especially useful for handling the rotation problem in our current case. On one hand, in the design of floor plans, most cells only rotate by a multiple of 90°. We do not have the burden to make our system recognizer cells of arbitrary degree of rotation and thus can use relatively fewer training samples per cell. On the other hand, the R block's sensitivity to noise might not be appropriate because that some extraneous entities such as marks for grids might appear in the drawing. Based on these observations, we developed a prototype system for cell recognition, which is described in the next section.

4 A Prototype System for Cell Recognition

We have implemented a prototype program for cell recognition in the Intergraph MicroStation System. The whole program consists of two phases. The first phase, ***the focusing phase***, determines where the user is currently working and captures the surrounding screen image. The second phase, ***classification phase***, then uses a neural network to determine what the user is drawing based on the captured image. The neural network returns the indices of two cells that are most similar to the input. The system then displays these two cells in another window and allows the user to select the suggested cells. The following two sections will describe these two phases in more detail.

4.1 The Focusing Phase

Our system allows the user to draw freely at any location in the design window. The focusing phase must therefore determine which group of drawing entities should be captured and fed into the neural network for classification. Obviously, the most recently drawn or modified drawing entity must be part of the object that the user is drawing. However, it is not trivial to determine whether the user is starting a new object or has drawn a portion of the object, and in the second case, what the other parts of the object are. Two characteristics, spatial and temporal proximities, associated with cell drawing processes provide us with a clue about how to approach this problem. First, cells tend to be localized and in smaller scale with respect to the whole drawing, which we consider as the property of ***spatial proximity***. Second, we have argued elsewhere[14] that a CAD user tends to complete the drawing of one object before starting on another object, which we consider as the property of ***temporal proximity***. Based on these two properties, we can maintain a window, both in time and space, to monitor the user's drawing process and determine where the user is focusing his or her drawing activity.

Using the property of temporal proximity, a window, W, is maintained to cover the last n entities drawn consecutively. The window size (n) is then decided by assuming the property of spatial proximity. When an entity is created or modified, a window, w, containing this single entity is also created. The spatial relationship between W and w determines how W should be modified to accommodate the newly created or modified entity in w.

We have categorized the spatial relationship between W and w into four types. First, if w is far away from W, we assume that the user is starting a new object and make w the new W. Second, if w is close to W or w is overlapped with W, we enlarge W to cover w. Third, if w is inside W and w's area is far smaller than W's (if w contains a

[13] In Yüceer and Oflazer's original work, noise was added to the characters on the foreground only, while our study added noise also to the background.

[14] Dershung Yang and Doris S. Shaw. Object recognition in the design environment: A neural network approach. In *Proceedings of the 6th International Conference on Systems Research, Informatics, and Cybernetics*, volume 1, Baden-Baden, Germany, 1992.

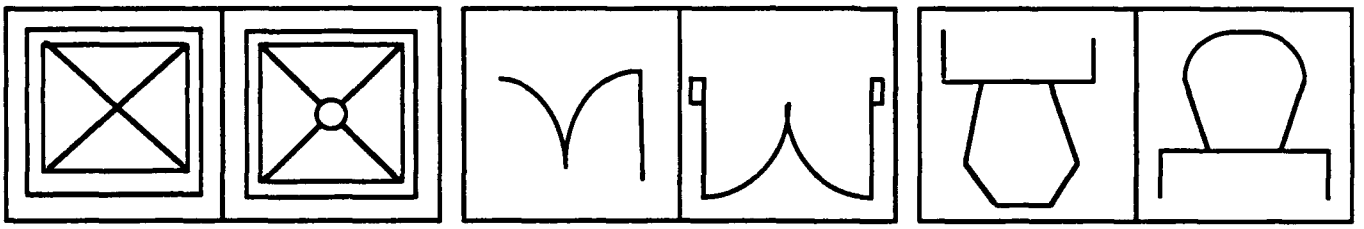


Figure 1: Examples of interactions between the user and the program. See text for details.

horizontal or vertical line, then the comparison is based on the length rather than on the area), we also assume that the user is starting a new object and let W be w. This situation typically happens when the user finishes the layout and begins to work on the details. Finally, if w is inside W but occupies a large area in W, then we make no change to W because the user might still be working on the same object.

After W is updated, the bitmap image in W is captured and fed into the neural network for the second phase.

4.2 The Classification Phase

The bitmap image captured in the focusing phase is normalized both in size and in location into the center of a 48×48 pixel input area, using Yüceer and Oflazer's ST blocks. (The R block is not used due to its sensitivity to noisy data. Rotations are handled by presenting 4 rotated images per cell, one for each 90°, as training examples.) The system then feeds the normalized image to the embedded neural network for classification.

The Zipcode Net (Le Cun, *et al.*) was chosen as the neural network architecture. The training examples consisted of 23 cells that are most commonly used in the design of architectural floor plans. With four different directions per each cell, there were total of 92 training examples. The network consisted of three hidden layers. The first hidden layer had 6 feature maps, each of size 15×15 pixels. The second hidden layer also had 6 feature maps, each of size 6×6 pixels. The third hidden layer consisted of 15 nodes. Connections between the output layer and the third hidden layer, and between the third hidden layer and the second hidden layer were fully connected. Each pixel in the second hidden layer received inputs from a 5×5 kernel from all the feature maps in the first hidden layer. Two kernels corresponding to two adjacent pixels were offset by 2 pixels. Each pixel in the first hidden layer receives inputs from a 6×6 kernel in the input layer. Two adjacent kernels were offset by 3. The total number of nodes was 3908 and the total number of free links was only 4701 due to the weight sharing and local connectivity techniques. The training process was identical to that described in section 3.2. The network converged after 1156 cycles.

Preliminary tests of this prototype system showed encouraging results. Figure 1 shows three examples of how the CAD user interacted with our system. In each pair of examples, the left rectangle shows the user's drawing and the right rectangle shows the cell identified by our program. (We only show the cell with the largest activation.)

5 Summary

In summary, this paper presents a practical use of neural network technology. Our prototypical system is embedded in a CAD system and intends to help the CAD user utilize standard cells effectively. Cells play an important role to facilitate communication in a cooperative environment. However, the number of cells is enormous and thus difficult to manage. Our system intends to lessen the difficulty by automatically identifying the cells that are similar to what the user is drawing and suggesting that the user replace the drawing with standard cells. This process is done while the user is drawing objects on the screen.

Transformation-invariant pattern recognition is the most important research issue for this system. We have studied and compared four different approaches appearing in recent neural network literature. Our results indicated that the Zipcode Net performed best compared to the Third-order Net and the Backpropagation Net. In a noisy environment, the Zipcode Net without RST-blocks performed surprisingly well. The RST-blocks were in general effective for removing the effect of transformations in the raw images. Nevertheless, the RST-blocks quickly deteriorated when noise was present. We believed that the major reason for the poor performance of the RST-blocks in a noisy environment was because of the R block. Therefore, our system employed the Zipcode net preceded by the ST blocks only and used several rotated images as training examples to solve the rotation problem.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

A Neural Network Approach to Fault Isolation

in

Units Under Test (UUTs)

17 July 1992

Paul V. Hayes and R. Timothy Rue
ITT Defense
Aerospace/Communications Division
Fort Wayne, IN 46818
(219) 487-6641
(219) 486-6648

Samir I. Sayegh
Purdue University
Fort Wayne, IN 46805
(219) 481-6880

ABSTRACT

This paper first discusses the present state of the art of TPS (Test Program Set) diagnostic approaches and then surveys the literature available on the use of neural networks for TPS diagnostics. Next, our approach to mapping the TPS diagnostic problem to a supervised Back Propagation neural network solution is presented. Techniques used to collect and pre-process the training patterns for the neural network are covered. Finally, simulation results for a simple sequential digital UUT are reported, and our plans for future research are outlined.

I. Introduction

Full diagnostic Test Program Sets (TPSs) require skilled engineering development effort. This paper describes an effort currently underway to partially automate the Fault Isolation/Diagnostic portion of this problem using neural networks.

A TPS is made up of the hardware (ICD or Interface Connecting Device), the software (test program Go-Chain and diagnostics), and the documentation required to test and maintain a UUT. The TPS is used in conjunction with Automatic Test Equipment (ATE). The TPS hardware interfaces the UUT physically and electrically to the ATE.

A set of UUT test stimulus vectors that are known to be effective in detecting faults are prepared beforehand (the Go-Chain). Faults are inserted into a UUT, the stimulus is applied to the UUT, and then UUT response vectors are collected. A compressed version of the resulting UUT stimulus/response vector set is then used as input to train a supervised neural network. Knowledge of which fault is associated with which vector set is used to generate the corresponding desired outputs for training the neural network. Once trained, the network takes as input the compressed UUT operational connector behavior and provides as output the fault or faults that are present in the UUT.

Potential advantages of a neural network solution include the following: No (or reduced) TPS operator probing for fault isolation; improved ability to deal with the presence of multiple faults; reduced TPS development time; reduced TPS Mean Time to Diagnosis (MTTD); improved ability to deal with the lack of UUT initialization, due to either the presence of faults or a lack of testability; and reduced TPS maintenance costs due to UUT design changes.

Currently the AN/USM-465A digital card tester is being used in conjunction with a simple sequential digital UUT. Initial results are encouraging for UUT1, using a two layer neural network and a readily available AN/USM-465A function for data compression. The fault universe considered to date is all possible, non-destructive Stuck-At-Zero (SA0) and Stuck-At-One (SA1) faults.

II. TPS Diagnostics, Current Approaches and Problems

Given that an effective test program or set of test stimulus vectors are available (the Go-Chain), the problem is then to determine if a UUT is good or faulty, and if bad, what component on the UUT is bad. Current fault isolation techniques make use of information available at the UUT's operational (edge) connector, as well as information collected by probing the internal nodes of the UUT. It is desirable to minimize or eliminate the amount of probing required. This reduces the need for UUT disassembly, operator positioning of the probe, piercing of the UUT's conformal coating, the subsequent need for repair of the conformal coating, and the potential for operator mis-probing and resulting UUT misdiagnosis.

A "Go-Chain" is typically used to determine if the UUT is good or bad. Once it has been determined that the UUT is bad, it is often necessary to use special diagnostic routines (stimuli as well as measurements that are not part of the Go-Chain) to isolate the faulty component(s). These diagnostic routines are often costly to develop and are prepared by an engineer based on an analysis of the UUT's circuitry, simulation of UUT behavior, and on empirical test (fault insertion) results.

There are several widely used Computer Aided Test (CAT) tools on the market for digital UUTs. CAT tools for analog TPS development are less useful. For very "testable" digital UUTs, the CAT vendor's canned, and typically proprietary, diagnostic algorithms can be used in conjunction with simulator-generated data to provide UUT diagnostics. By "testable," it is meant that the UUT is more or less compatible with the ATE. One very important measure of testability, common to all brands of ATE, is the ability to quickly bring the UUT's internal state storage elements to a known state: initialization.

Even if it is possible to initialize a good UUT, it is likely that there are several UUT failure modes that prevent initialization. When initialization is not possible, the UUT might respond differently from one run (application of the test program to the UUT) to another run. Accounting for this varied behavior in the test program can be very difficult and costly.

Fault dictionary approaches to TPS diagnostics, which require less TPS development time, typically run an order of magnitude faster than guided probing approaches. Using a fault dictionary approach amounts to making note of where (which edge pins) and when (what test step) a particular fault is first detected in the Go-Chain and hard coding an appropriate fault callout response. However, due to the initialization problem it is often not practical to use a fault dictionary or look-up table.

For modern complex UUTs the number of test steps in the test program can be in the millions. Storage of UUT node vectors can thus be demanding. Another ATE approach, implemented in the AN/USM-465A, is to compress the node/state data into a signature. Various proprietary techniques are used for this operation. For the AN/USM-465A, two six digit signatures are used per node. A known good UUT is required to generate the signatures. This data, combined with a topological map of the UUT (component input to output relationships are encoded using component reference designators), is then used along with a proprietary guided probing algorithm to isolate faults.

A common simplifying assumption is that there can exist only one fault in the UUT at a time. In the case of TPSs developed for the US Army, TPS acceptance testing is typically limited to the insertion of a single fault. This assumption is made to keep the costs practical for Go-Chain and diagnostic routine preparation.

Another simplifying compromise is to discontinue the fault isolation process prior to indicting only a single component. It is sometimes not possible or practical to isolate to a single component, even when only a single faulty component is present in the UUT. In cases like this it is customary to isolate to an ambiguity group of components, any one of which may be the single faulty component. This approach is used for US Army TPSs. A sampling of 608 fault callout messages from 23 different fielded US Army SINCGARS AN/USM-465A TPSs indicates that the average number of components per call-out is about 2.4. As the average component gate count and complexity increases this approach is rapidly becoming less viable.

A review of the literature revealed previous work in the area of TPS diagnostics using neural networks. Reeder and Koos [1990] gave no quantitative results and indicated that they had decided to avoid sequential digital circuits for their first experiments and stick to combinational logic only. They also expressed a preference for using simulated fault insertion over actual hardware fault insertion. In addition, Reeder and Koos indicated that it was important to take into account historical component reliability data when selecting the types of faults for generation of backpropagation neural network training patterns.[1]

Jakubowicz and Ramanujam [1989] reported developing a capability to direct or assist a technician during the fault isolation process using a neural network approach based on Kohonen feature maps and the delta rule. Their network then showed an ability to identify the correct fault or propose the next test. In this approach, UUT specific fault/symptom information is presented to the network, as are some basic search procedures. The network is taught, for example, the binary search process, or search schemes based on component reliability.[2]

In contrast to the work done by Reeder and Koos, we use hardware fault insertion, and a UUT with a memory storage element. In contrast to the Jakubowicz and Ramanujam [1989] approach, we use only a supervised backpropagation neural network and make immediate fault callouts.

III. Mapping the TPS Diagnostic Problem to a Neural Network Solution

The main problem anticipated by the authors when first considering a neural network solution to TPS diagnostics was the sequential nature of UUTs. A typical real world UUT may require millions of stimulus

---

[1]J. R. Reeder & L. J. Koos, "A Neural Net Approach to Electronic Circuit Diagnostics", Proceedings of IJCNN, January 1991, vol. II, P. 671-674.

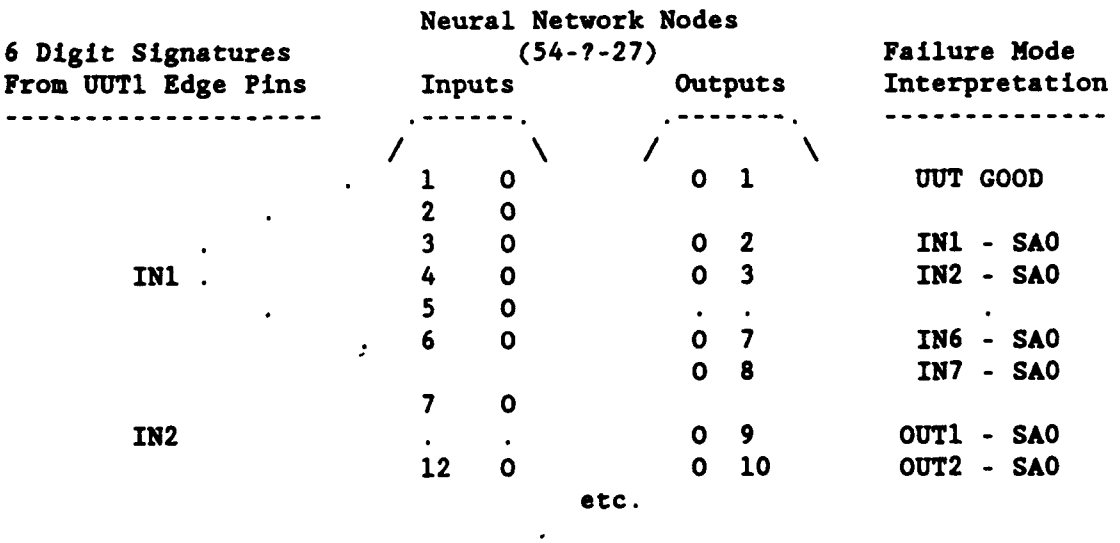
[2]Jakubowicz O. and Ramanujam S., "A Neural Network Model for Fault-Diagnosis of Digital Circuits", Proceedings of IJCNN, January 1991, vol. II, P. 611-614.

vectors for complete testing, and thus millions of response vectors. The order of application of the stimulus vectors to the UUT is also critical. Typical approaches to problems of this nature are to adjust the sampling rate of the data to as low a rate as feasible, and then to "bin" samples by applying some type of compressing transformation to the data. In our approach, UUT nodal states are sampled after every test step. Edge node behavior is compressed into a single six digit signature (CRC-16). This was particularly convenient since the AN/USM-465A has this data compression capability built in.

For UUT1 we chose a simple neural network architecture and paradigm. We use a supervised backpropagation paradigm. After experimenting with the number of hidden layers and the number of nodes per hidden layer, we obtained good performance with only two layers (i.e. no hidden layers) and a linear transfer function. We do, however, expect that more complicated UUTs will require correspondingly more complicated architectures.

The six digit signatures from each UUT edge pin are mapped directly to six neural network input nodes. Each UUT failure mode considered (i.e. each SA0 and SA1 fault) is mapped to its own output of the neural network. For two layer, linear transfer function, neural network training it was not necessary to normalize the input data which ranged from 0 to 9. For three layer, non-linear training each digit of the UUT six digit signature was divided by 10 for compatability with the hyperbolic tangent transfer function.

The desired outputs used to train the networks were set at +1 to indicate the presence of a particular fault (or of a good UUT) and to -1 for the rest of the faults not present when the UUT edge signatures were generated. There are a total of nine edge pins on the UUT (7 inputs and 2 outputs). Both UUT input and UUT output pin signatures were used as input training patterns to the network for the majority of the experiments conducted. Thus there were a total of 54 (9 x 6) neural network inputs. UUT1 has a total of four internal nodes (TP1 through TP4). These four internal nodes, combined with the nine edge nodes, make for a total of 13 nodes for fault insertion. Inserting both SA0 and SA1 faults on all 13 nodes, allowed for a total of 26 faults. Thus the network was constructed to have 26 output nodes, plus one output node to represent a healthy UUT giving a total of 27 output nodes. This mapping of the faults and output values allowed for easy interpretation of the network's response. The output with the largest or most positive value was interpreted as being the network's indication of which fault was present.



Mapping of the TPS Diagnostic Problem to a Neural Network Solution

IV. Simulation Results

A. Training Results (2 Layer vs. 3 Layer)

100% performance was achieved with both two and three layer networks. As it turned out, we found that there were only 22 faults that produced unique edge pin signatures for UUT1. Four of the faults (26 - 22 = 4) produced redundant UUT1 edge pin behavior. This was discovered when trying to understand some initial poor network performance. As a result a software utility was developed which screens for duplicate input-output pattern pairs. When two different faults inserted into the same UUT produce identical edge pin signatures, this will produce a contradicting case for the network. In this case, the inputs to the network are equal, but the network is expected to produce two different output responses. The table below summarizes the pattern types screened by our utility.

Inputs Equal	Outputs Equal	Type of Input-output Pattern Pair
No	No	Typical Training Patterns
Yes	No	Contradicting Patterns
No	Yes	Classification Challenge
Yes	Yes	Identical/Redundant Patterns

Neural Network Training Pattern Types

B. Initialization Test Results

The test program for UUT1 was rewritten in order to test the key issue of the network's ability to deal with a lack of UUT initialization. For this test, the latch of UUT1 was manipulated to both of two possible states for edge pin signature collection. Each of the 22 faults were inserted after starting the latch's q-line out at a logic low. Fault insertion was then repeated while starting the q-line at a high level. As could be expected, several of the faults, when inserted into UUT1, removed the pre-programmed initial condition from the latch. However, it was found that a total of 14 of the 22 faults inserted produced different edge pin signatures, depending on the power-up state of the latch (i.e. classification challenges). We trained the network with data generated when the q-line started out low and tested the network with data generated when the q-line started out high. The training performance reached 100% within a few hundred epochs (same order each time and sequential weight updating was used). As mentioned above, eight of the 22 test patterns were identical to eight of the training patterns, due to the faults removing the initial conditioning of the latch. The network successfully generalized for 12 of the remaining 14 patterns. One of the two missed patterns was for UUT good behavior. The other was a flat out miss by the network. Since we have the Go-Chain, we do not have to rely on the network for determining whether or not the UUT is good or bad. Therefore we scored this test as being about 92% (12/13) successful. In other words, for a very simple UUT our network was able to correctly generalize in order to handle the initialization problem 92% of the time.

C. Two Fault Test Results

The setup for this experiment is to train with UUT edge pin signatures generated from the insertion of a single fault and test the network with signatures generated when two faults are inserted into UUT1. We were only able to insert a total of 20 fault pairs. Two of the 20 patterns were found to contradict other patterns, and were eliminated. For inputs to the network generated with two faults present in UUT1, the two largest actual outputs from the network are interpreted to be the two faults indicted by the network. This experiment is still in process.

D. SA0 vs. SA1 Test Results

To test for the network's ability to generalize across fault types at a given node, the network is trained with UUT edge pin signatures produced from SA0 faults and then tested with SA1 patterns. Although this effort is still in process, initial results look promising. We expect to get results comparable to those had during the test for initialization.

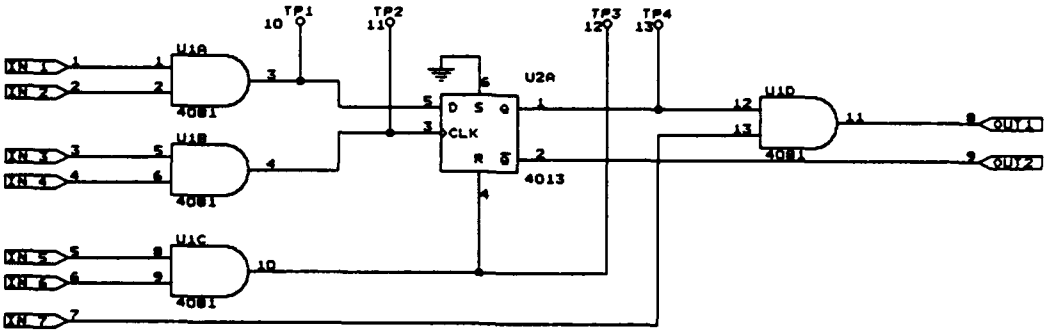
E. UUT Outputs Only Test Results

For this experiment, the patterns from the initialization test were modified. The signatures in all network input patterns corresponding to all UUT input edge pins were eliminated. The network's architecture was altered correspondingly. A total of only 12 input nodes remained (reduced from 54). Six of these correspond to each of UUT1's two output edge pins. This effort has not yet been completed.

V. Conclusions and Future Directions

Although we are presently working with a very simple UUT, we are encouraged by the fact that simple network architectures are handling some of the basic TPS diagnostic issues. We feel that our approach will scale sufficiently to provide a useful compliment to present diagnostic development techniques, for more complex UUTs. Future plans include: investigate the relative effectiveness of various methods of compressing the training data; study the relative effectiveness of different network architectures at generalizing successfully in the presence of previously unseen input test data; work with more complex digital UUTs; and work with analog UUTs.

UUT1 schematic



Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

APPLICATIONS OF COMPUTER ALGEBRA SYSTEMS TO NEURAL NETWORKS

Samir I. Sayegh
Physics Department
Purdue University
Ft Wayne, IN 46805
sayegh@ecn.purdue.edu

ABSTRACT

Computer Algebra Systems (CAS) have been growing in power and sophistication extending by several orders of magnitude our ability to develop both exact solutions and expansions in terms of a parameter. The availability of stable and flexible CAS has allowed the Scientific and Engineering communities to tap into such systems with impressive results.

We present three examples of the use of CAS in the field of Neural Networks: 1) the automatic generation of network code from general specifications of a network, 2) the inheritance of weights from one network to another and 3) an expansion of the difference between sequential and cumulative update strategies for the Back Propagation (BP) algorithm in terms of the learning rate.

Such techniques should be viewed in a unified way and the examples provided are illustrative of the myriad possible applications of CAS for Neural Networks.

1. INTRODUCTION

The first and most straightforward application is the use of GENTRAN for generating C or FORTRAN code from a general specification of a neural network in REDUCE, allowing great flexibility in network design. This can be done, for example, by specifying different error functions and nonlinear "squashing" functions and automatically generating the numerical code to implement the corresponding network.

The second application is that of the inheritance of weights from one network to another. A fast learning but less than optimal network is trained first. Inheritance equations, derived via symbolic algebra, are applied to transfer the weights to the slower but optimal network, thus completing the training phase. Alternatively, for networks where learning and feed-forward phases have different > 1 and < 1 speeds as compared to a standard, the training can be implemented on the faster learning network while the feed forward (operation) phase is left to the other. A typical example is that of the Probabilistic Neural Network (PNN) [12] when compared to Back Propagation (BP).

The third and final example arises in a number of situations, where expansions in terms of a parameter are appropriate for a neural network. The learning parameter in an adaptive system is usually a "small" number. It is also possible to write addition formulae for the sigmoid or hyperbolic tangent, which when coupled with their $< |1|$ range, result in expansions that preserve nonlinear properties while retaining only a few terms. Such expansions can be carried out by the use of CAS. An example of such techniques is presented where the difference between sequential and cumulative update strategies for BP is expanded in terms of the learning rate.

2. AUTOMATIC GENERATION OF NETWORK CODE

In feedforward networks of the backpropagation type, the number of hidden nodes, the nature of the squashing function, the error function and other parameters are usually determined experimentally for lack of a firm theoretical framework to optimize their choice. While theoretical work towards developing such a framework is ongoing, many researchers find themselves rewriting code appropriate to a variety of situations. The use of symbolic languages can alleviate the problem. The example we will discuss is that of the code generation system GENTRAN running under the computer algebra system REDUCE [7].

A simple example of code generation and the resulting C and FORTRAN code is given here. The following code fragments are written in REDUCE and the function eval, evaluates the expression of its argument. Here both the nonlinear squashing function and the error function are left arbitrary and their subsequent definition is unfolded in the numeric code. Note that the necessary derivatives for the backpropagation phase can be generated likewise.

```
load "gentran"$
gentranlang!* := 'c$ COMMENT LANGUAGE IS C $
on getdecs$ COMMENT AUTOMATICALLY GENERATE TYPE DECLARATIONS$
operator f$
for all x let f(x)=tanh(x)$ COMMENT CHOOSE A TRANSFER FUNCTION HERE$
for all x let funcerror(x) = x**2$ COMMENT CHOOSE AN ERROR FUNCTION HERE$

gentran
for h:=1:nHidden do
<<
 nethid(j):= for i:= 0 : nInput sum u(h,i)*x(i)$
 hid(j):= eval(f(nethid(j)))$
>>$ COMMENT TRAVERSING FIRST LAYER$

gentran
error:= for o:=1: nOutput sum eval(funcerror(des(p,o) - out(o)))$
COMMENT COMPUTING THE ERROR$
end$
```

3. INHERITING KNOWLEDGE IN NEURAL NETWORKS

It is important to develop techniques to transfer "knowledge" between two networks. Some previous work has been done in this direction by inclusion of non linear elements [2,3], modular approaches that are later "glued" [13,4,5], and taking advantage of the principle of Least Action [8,9]. Here we take a direct approach to the question of inheritance. Networks S (source) and T (target) are implementing two different mappings $S(w_s,x)$ and $T(w_t,x)$ respectively, where x is the input vector and w_s and w_t represent the respective weight vectors. In order to derive the inheritance equations, one writes an expression of the form

$$J(w_s,w_t) = \int \rho(x)\, d[S(w_s,x) - T(w_t,x)]dx$$

where J is to be minimized with respect to the w_t. d is a distance measure between the outputs and ρ represents a distribution density. ρ could simply represent a distribution such as that of the training patterns and/or a weighing factor such as a factor favoring large errors where large dollar amounts are involved in a financial application. Minimization of J is achieved by requiring the derivatives with respect to w_t to vanish. Two situations of interest arise. The first is where the source network S achieves a suboptimal classification with fast convergence while the target network T reaches optimal (or at least better) classification with slower convergence. One would train S first, transfer its "knowledge" to T and proceed with training T. The second situation where such inheritance is fruitful is that where S trains faster but T executes (feeds forward) faster, without necessarily having different performances. Training with S and transferring the weights to T for execution is desirable. For example, claims have been made that in some applications, Probabilistic Neural Networks [12] train orders of magnitude faster than Back Propagation [6] while executing about one order of magnitude slower.

As an application consider the problem of feedforward networks. We start with a two layer network with fast but limited learning capability. Since three layers are sufficient [1], we attempt to inherit weights from the two layer net to a three layer net. An arbitrary number of hidden units, H, and, the same number of input units, I and output units O as those of a two layer net are to be used. The weights in the first weight layer are denoted by u^1_{ih}, for weight connecting input node i to hidden node h. Similarly, the weights connecting hidden node h to the unique

output node is denoted by v^h. The nodes of the two layer net are denoted by wi. The problem can now be stated as follows:

"For a given two layer network, having been trained and resulting in weights w^j, i=0, ..., I-1, find weights u^i_h and v^h, i=0, ..., I-1 and h_i=0, ..., H-1, such that the resulting three layer net best approximates the performance of the two layer network in the LMS sense."

Setting:

$$A_{ij} = \Sigma f(u_i^r x_r) x_j$$

$$B_{ih} = \Sigma f(u_i^r x_r) f(u_h^i x_i)$$

$$C_{rmj} = \Sigma x_r f'(u_m^s x_s) x_j$$

$$D_{rmh} = \Sigma x_r f'(u_m^s x_s) f(u_h^i x_i)$$

For known w^j, one derives the following equations (I×H of them)

$$[C_{rmj} - D_{rmh}(B^{-1})^{hl} A_{ij}] w^j = 0$$

representing I×H (nonlinear) algebraic equations in the I×H unknowns u_m^n.
Once these equations are solved numerically, one can use

$$(B^{-1})^{hl} A_{ij} w^j = v^h$$

to determine the v's. Notice that equations for the unknown weights u's and v's are decoupled, and that the v's are essentially solved for, provided B can be inverted. Programs such as REDUCE, MACSYMA or MATHEMATICA [7] can be used both to write the required expressions and to solve the equations. The inversion of the B matrix is not problematic since B is an H dimensional square matrix and H is typically smaller than I. Resulting equations for the u's can become involved. These are, however, polynomial equations since there exists a polynomial relation between the squashing function and its derivative as well as rational relationships between the squashing function of a sum and the squashing function of individual terms in the sum. For example, for the hyperbolic tangent the following relationships hold:

$$\tanh'(x) = 1 - \tanh^2(x)$$

$$\tanh(a+b) = \frac{\tanh(a) + \tanh(b)}{1 + \tanh a \ \tanh b}$$

Similar equations are easily derived for the sigmoid. Such relationships, allow for writing polynomial equations in the hyperbolic tangent of the individual weights. This in turn, allows the use of Symbolic Algebra programs,

as they are well suited to polynomial manipulation, including truncation of higher order terms in search for approximate solutions. The truncation of higher order terms originates from the fact the hyperbolic tangent is always smaller than one and the product of several such terms becomes negligible in comparison to an individual term.

The above approach applied to XOR and parity problems results in significant speed up factors [10]. A number of advantages should be noted.
- Results are problem independent in the sense that *no* output information is specified. This makes a symbolic solution very attractive.
- Some local minima can be side stepped, avoiding non optimal solutions and contributing further to faster convergence as the number of restarts is decreased.
- In the two-to-three inheritance, equations for u and v are decoupled. The equations for u, though nonlinear, are amenable at least to an approximate solution. In some cases as in the $2\times2\times1$ net, they simply become identities.

4. A SERIES EXPANSION FOR SEQUENTIAL-CUMULATIVE UPDATES

This last example of the use of CAS, develops an epsilon expansion of the difference in the values of the weights at the end of an epoch, when sequential and cumulative updates are used. Different claims are made in the literature as to the merit of the different updating schemes. We have shown that, in first order in the learning rate, epsilon, there is no difference at the end of an epoch between the two schemes. The development of the result in its more general form is based on the availability of Computer Algebra Systems, to allow for the required expansions in the nontrivial cases [11].

5. CONCLUSION

Computer Algebra Systems can play a crucial role in the area of Artificial Neural Networks. This has been illustrated in solving both theoretical and very practical problems. Despite the general perception that neural networks are generalized *numerical* algorithms, the use of *symbolic* tools is a powerful complement to numerical techniques. The availability of flexible systems and powerful platforms make CAS a practical tool for current scientific and engineering applications.

REFERENCES

[1] Cybenko, G. 1988. Continuous Valued Neural Networks with Two Hidden Layers Are Sufficient. Technical Report, Department of Computer Science, Tufts University, Medford, MA.

[2] Giles, C.L. and T. Maxwell. 1987. Learning Invariance and Generalization in Higher Order Neural Networks. *Applied Optics*, vol. 26, pp. 4972-4978.

[3] Pao, Y.H., and R.D. Beer 1989. The functional-link net: A unifying Network Architecture Incorporating Higher Order Effects. INNS First Annual Meeting, Boston, IEEE.

[4] Pratt, L.Y., and J. Mostow 1991. Direct Transfer of Learned Information Among Neural Networks. *Proceedings of AAAI-91*, Morgan Kaufmann.

[5] Pratt, L.Y. and C.A. Kamm 1991. Improving a Phoneme Classification Neural Network through Problem Decomposition. *Proceedings of IJCNN-91*, IEEE.

[6] Rumelhart, D.E., G.E. Hinton, and R.J. Williams 1986. Learning Representations by Back-Propagating Errors. *Nature* 323, 533-536.

[7] Raynha, G. 1987. *REDUCE, Software for Algebraic Computation.* Springer Verlag.

[8] Sayegh, S.I., 1989. Fingering for Strings Instruments with the Optimum Path Paradigm, in *Music and Connectionism*, MIT Press.

[9] Sayegh, S.I., 1990. Learning with the Optimum Path Paradigm. *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C.

[10] Sayegh, S.I., 1992. Inheriting Knowledge in Neural Networks. *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C.

[11] Sayegh, S.I., 1992. Sequential vs Cumulative Update: an ϵ Expansion. *Proceedings of the Fifth Conference on Neural Networks and Parallel Distributed Processing*, Fort Wayne, IN.

[12] Specht, D.F., 1988. Probabilistic Neural Networks for Classification, Mapping, Or Associative Memory. *Proceedings of IEEE International Conference on Neural Networks*, San Diego, Ca.

[13] Waibel, A. 1989. Modular Construction of Time-Delay Neural Networks for Speech Recognition. *Neural Computation* 1, 39-46.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

Membrane Equation and Implementation of Neural Networks

Bahram Nabet

Electrical and Computer Engineering Department
Drexel University
Philadelphia, PA 19104

Abstract

In this paper, ionic conduction in excitable cells, the well-known membrane equation, is shown to lead to simple analogue electronic circuitry with rich processing capabilities. The paper starts from simple components and demonstrates increasingly complicated processing by fusing the components together.

1.0 Introduction

The classic Hodgkin-Huxley[1] model of ionic flow in nerve membranes is shown in Figure 1 and described by the equation

$$C\frac{dV}{dt} = (V^+ - V)\,g^+ + (V^- - V)g^- + (V^P - V)g^P \qquad (1)$$

where C is the membrane capacitance; V^+, V^-, and V^P are excitatory, inhibitory, and passive saturation points, respectively; and g^+, g^-, and g^P are excitatory, inhibitory and passive membrane conductances, respectively.

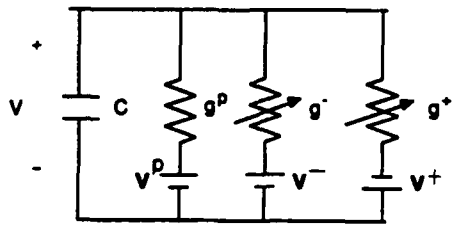


Figure 1. *An isopotential patch of membrane. Batteries show Nernst potentials; g^+, g^-, and g^P are, respectively, passive, excitatory and inhibitory conductances.*

[1]Hodgkin, A. L. and Huxley, A. F. (1952). "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Journal of Physiology*, 117, 500-544.

Many neural network models can be derived by assigning variables to Equation (1). For example, letting the Nernst potentials (modeled by the batteries) to be zero, ignoring the excitatory ionic channel g^+, and considering the case where the inhibitory conductance is controlled by the voltage of other cells, that is, $g^- = \Sigma_j f_j(x_j)$, we obtain

$$\frac{dx_i}{dt} = I_i - A x_i - x_i \sum_{j \neq i} f_j(x_j) \qquad (2)$$

where the capacitance value is set to 1, I_i are (excitatory) current inputs to the cell, and A is the passive membrane conductance. This is a multiplicative lateral inhibitory feedback neural network. The multiplicative nonlinearity arises from conductance modulation and endows such networks with important computational capabilities.[2] It is a straightforward matter to show that feedforward, or nonrecurrent versions of neuronal interactions are obtained when the conductances are modulated by inputs, rather than the activity of other cells as is the case in (2).

The Hodgkin-Huxley equation thus provides the crucial insight for the electronic circuit designers that models of neural networks can be constructed from electronic devices or circuits whose conductance may be varied as a function of voltage or current values. The simplest such device is a single Field-Effect Transistor (FET) which, when operated below pinch-off, acts as a voltage-controlled conductance. The next sections show feedback and feedforward networks which can be built from these elements.

2.0 Feedback Neural Networks

Figure 2a shows a simple electronic analogue of the biological model of Figure 1. The inhibitory conductance is an FET whose conductance may be modulated by the voltage present at its gate. If this voltage is applied by other cells of the network layer, feedback interaction results; if this voltage is controlled by the previous layer, feedforward results. Figure 2b shows a recurrent network of four cells, connected to every other cell but not to itself, in a cross-bar layout. It is seen that each multiplicative interconnection is economically implemented by one transistor. Such implementations are truly analog in that the device physics performs the "computation" and no digital to analog conversion, or vice versa, is required.

Using analytical models for operation of FETs in the linear region, it can be shown that[3] for small values of node voltage the circuit of Figure 2b is described by Equation (2). This mathematical description also allows the network to be analyzed and its compatibility with different architectures, its stability, and classification properties examined.[4]

Response of a network of 32 cells, connected to their nearest neighbors, to a point source stimulus located on cell number 17 is shown in Figure 3. In this figure, the ratio of the stimulus to its surround has been kept at 2:1 but the mean input value has been changed to study the effect of varying mean light intensity on reflectance processing. The characteristic Mexican hat response is clearly observed at high mean intensities, but the inhibitory influence is reduced with decreasing available total input intensity. Similar change of the receptive field can be observed in many biological units including the DMC cells of the fly *lucilia sericata* as shown in the inset of this figure. A detailed analysis of field adaptation as a function of

---

[2]For a review see Grossberg, S. (1988)."Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," *Neural Networks*, vol. 1, 17-61.

[3]Nabet, B., Darling, R. B., and Pinter, R. B. (1989). "Analog implementation of shunting neural networks," in *Advances in Neural Information Processing*, Vol. 1, D. Touretzky ed., 695-702.

[4]Nabet, B. and Pinter, R.B. (1991). *Sensory Neural Networks: Lateral Inhibition*, Boca Raton: CRC Press.

mean input intensity in visual systems can be found elsewhere.[5] Other properties of this network include contrast enhancement, as apparent in Figure 3, and dynamic range compression.[6]

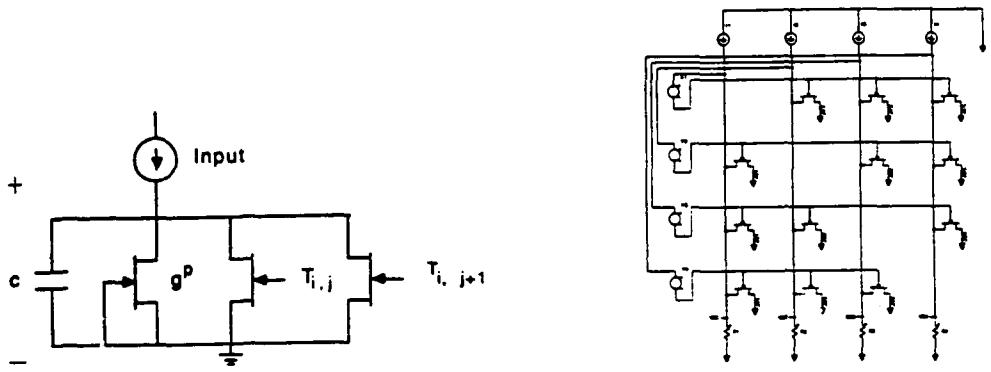


Figure 2. *a) Electronic circuit analogue of the membrane model of Figure 1 using FETs as variable conductances; b) A four cell feedback network.*

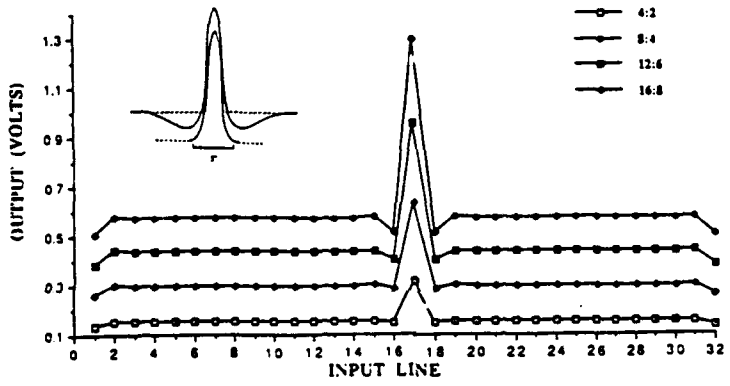


Figure 3. *Response of a 32-cell network, nearest neighbor connected, to a point-source.The ratio of the contrast of the point source with respect to its surround was kept constant while mean intensity was decreased from top to bottom. Inset shows similar response of the DMC cells of the fly.*

3.0 Feedforward Networks

As noted above, application of a voltage from previous layers of processing to the membrane analogue of Figure 2a produces multiplicative, or shunting, feedforward networks. A network of four cells with self-excitation and nearest neighbor inhibition is shown in Figure 4. The top portion of this circuit is the familiar crossbar architecture which defines the connectivity pattern of the network. The input lines are shown as voltages and are also available in inverted form to allow both excitation and inhibition. The lower array of transistors performs multiplication and nonlinear transformation of the input.

The circuit of Fig. 4 can be approximately described by

---

[5]Pinter, R. B. and Nabet B. (1991). "Field adaptation in visual systems is a function of nonlinear lateral inhibition", in *Neural Networks: Concepts, Applications and Implementations*, vol. 3, P. Antognetti and V. Milutinovic, eds., 21-48.

[6] Details appear in Nabet, B., Darling R.B., and Pinter, R.B. (1992)"Implementation of Front-End Processor Neural Networks," Neural Networks, Vol. 5, in press.

$$\frac{dx_i}{dt} = a_i - b_i x_i - x_i \left(\sum_{j \neq i} V_j / R_j \right) + c_i x_i V_i^2 \qquad (3)$$

where the capacitance is taken as 1, V_j's are input, x_i is output of cell i and a_i, b_i, and c_i are constants dependent on device characteristics and circuit parameters. The term in parentheses in the right hand side of (3) shows the desired multiplicative effect which causes an automatic gain control that allows the large input dynamic range to be processed by the network.

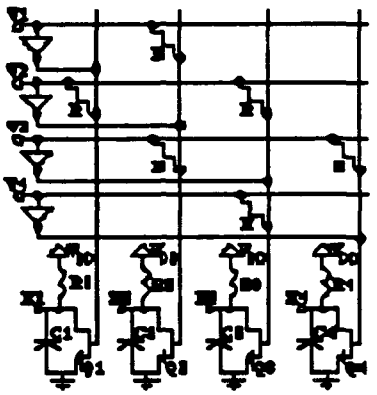


Figure 4. Circuit implementation of a shunting feedforward network. Nearest neighbor connections and self-excitation is shown for four cells.

A network of 80 cells with self-excitatory and two nearest neighbor inhibitory connections was simulated in PSPICE. A staircase input pattern ranging from 2 to 12 volts with 1 volt steps was presented to the network. The output of each neuron was limited to almost one tenth of the input range. The results of the implementation are shown in Figure 5. The output has the distinctive pattern of lightness perception in psychophysical experiments wherein edges are enhanced while uniform areas are attenuated. The network has also represented aninput range which is an order of magnitude larger than the dynamic range of each neuron and at the same time has preserved and enhanced the edges which contain the most information in the image.

Figure 5b shows an identical network which is operated in a region of transistor operation which does not have the multiplicative effect. Although both networks are allowed similar range of activity, at lower and higher values of input, assigned output levels are barely distinguishable in the network without gain control.

4.0 Motion Detection

The connection strengths of the networks described above are hard-wired. Addition of a second gate to the transistors of Figure 2a is a simple means of acquiring programmable connection strengths. The limit of programmability is a bi-level choice. This can be used in the networks described above to produce a unidirectional receptive field, that is, unidirectional shunting inhibition. This profile has direct application in determination of the direction of a moving image. It is known that both functional and anatomical models of motion detection operate on the same principle of nonlinear asymmetric interaction between channels from two adjacent receptor regions. In the case described above asymmetric interaction has multiplicative nonlinearity resulting from shunting inhibition.

An elementary motion detector can be constructed comprising of two subsystems each unidirectionally connected, but in the opposite direction, and competing in an opponent fashion, a simple form of which is subtraction. Response of the detector to a bar stimulus is shown in Figure 6 and is seen to be markedly different for motion is opposite directions. Response to drifting sinusoidal gratings and

moving edges are also distinct for motion in opposite direction.[7] The latter is interesting since there exist four possibilities: two polarity of contrasts moving in two directions. It can be shown that the response of the network is different in all four cases although ambiguous if direction is deduced only based on a thresholding operation.

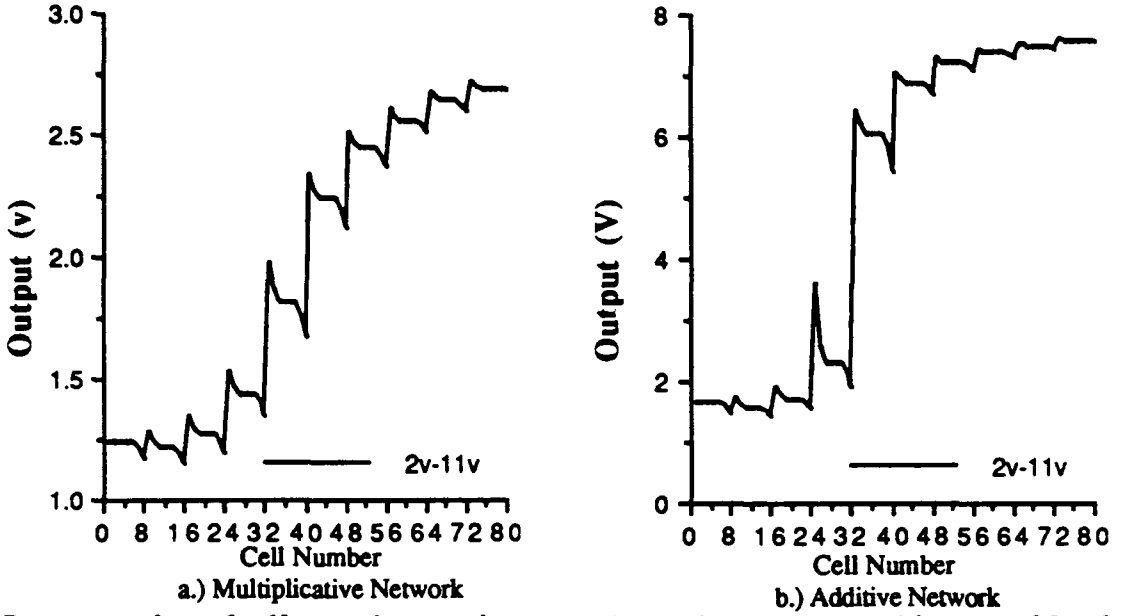


Figure 5. *Response of two feedforward networks to a staircase input pattern with steps of 1 volt. Network (a) operates on multiplicative inhibition while (b) has additive inhibition.*

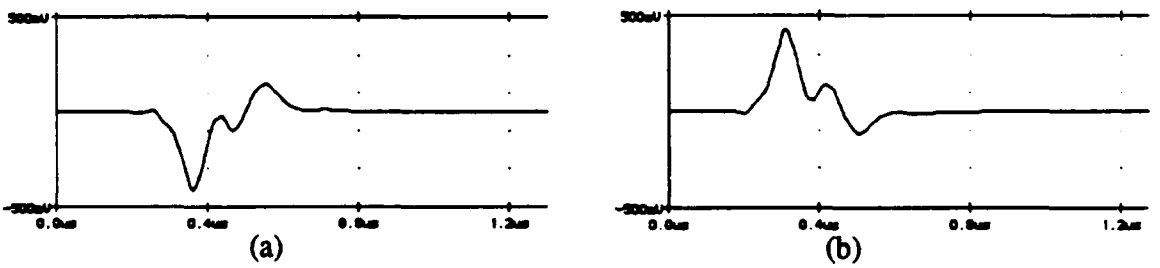


Figure 6. *Response of the shunting motion detector to a bar pattern moving in opposite directions.*

5.0 Conclusions

Hardware circuits described here are inspired by the basic membrane equation which suggests synaptic conductance modulation as a mechanism for achieving shunting feedback and feedforward network interaction. The circuits use the basic property of single field effect transistors which, in below pinch-off operation, are voltage controlled conductances. A circuit comprising of feedback networks and unidirectional synapses was shown to respond selectively to motion in opposite directions.The inherent fault tolerance due to parallelism, and the analog operation allows these operation to be performed with few devices and at high speeds.

Acknowledgement. This work was supported in part by the Engineering Foundation through Air Force Engineering Research Initiation Grant RIB9115.

[7] For results and analysis see V. Shreesha and B. Nabet (1992) "Modeling of a shunting movement detector, " Proc. IASTED Int. Conf. on Control and Robotics, August 4-6, Vancouver, Canada.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 **GACIAC PR 92-02**

Oscillatory Neural Networks on Multiple Limit-Cycle Self-Oscillators

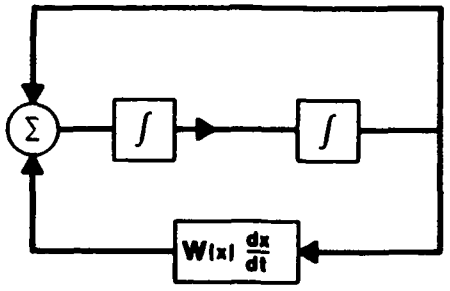
6 July 1992

By Y. A. Saet

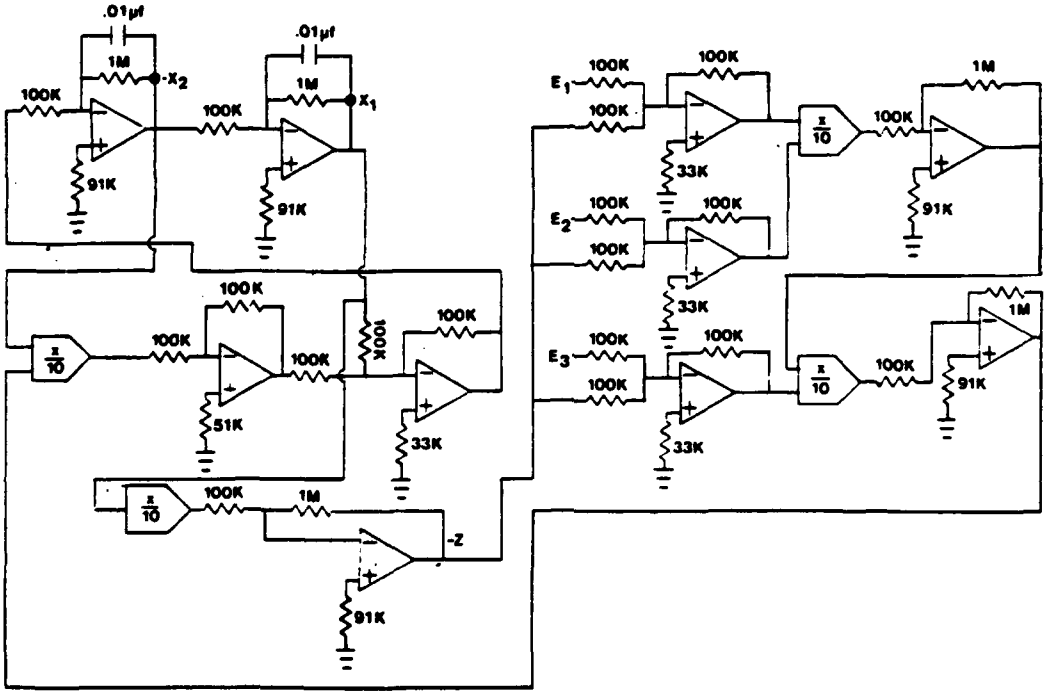
I. Introduction

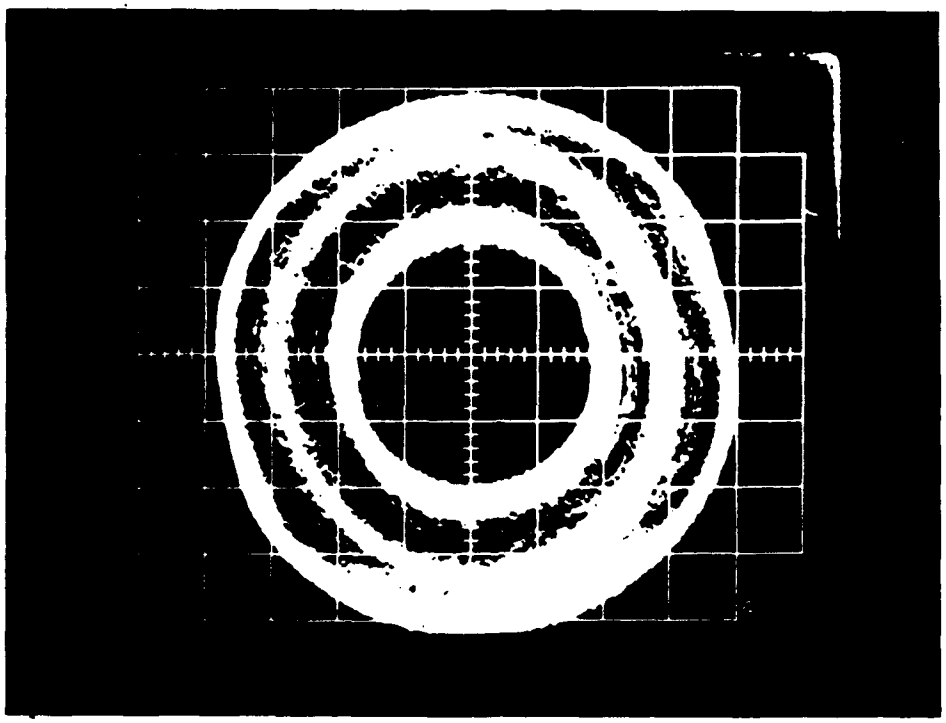
The paper is aimed at investigating the qualitative dynamics of self-oscillators with several limit cycles in relation to possible applications. We shall discuss both the hardware device-like aspect of the dynamics in connection with neural (broadly understood) networks and other quasi-biological objects, and also the concept itself as related to statistical mechanics of non-conservative dynamical systems and applied physics. Since the author (along with his collaborator) has undertaken systematic efforts to actually design, build, and outline the potential applications of multiple regimes electronic oscillators, the ensuing discussion will not refer to the already well-understood aspects of the subject matter. On the other hand, since the author is the only worker who has addressed the topic in any pragmatic manner, I shall allow for myself a few remarks of a historical or comparative nature. It was known from the days of Van der Pol and Andronov that a self-exciting oscillator might have more than one stable periodical regime. At that time, a possibility of this kind was viewed as an exotic event, chiefly due to a very limited control over the shape of the current-voltage characteristic of the electronic tube. For this latter reason, the only firmly established element of the concept of multiplicity inherited from that period is the notion of a hard mode excitation (two limit cycles but only one stable together with stable equilibrium). The last thirty years witnessed a significant widening of interest in our topic among other researchers. Among them are R. Landauer and his co-workers at IBM, who have consistently emphasized (for a very long time) the applied role and significance of various types of concrete dissipative physical systems with multiple states of stationary motions (which could be equilibriums or periodical orbits). To the best of this author's knowledge, since the mention of the possibility for the existence of several limit cycles in the self-exciting ondulators in the early thirties, up until 1984, there were no attempts to elaborate the topic or to obtain a real physical object with distinguished levels of excitation. Only recently the author and his coauthor have realized the self-exciting circuits with a number of limit cycles in each degree of freedom, and shown that such a circuitry can be built by the standard methods of production of the integrated components, including VLSI technology. The next immediate question then arises: From which point of view are the author's above efforts justifiable, considering the expectations of their possible application? I shall proceed with consolidating my position on the matter after the following remarks. There is an ocean of remarkable results which have been received during the last fifteen years in the areas of non-linear mechanics, oscillations, and their endless ramifications in the applied sciences. Among these

results virtually none was acquired exclusively by the means of the numerical experiments (perhaps the iterations of the mapping on the unit interval are the exception). In most of the cases, the discoverers of the new and interesting phenomena used simulation devices for getting the first overall qualitative pictures, which then were confirmed, complemented, and clarified by numerical results. This situation is often true even when the authors are eventually able to receive the formal proofs and theorems of an adequate character relative to the nature of the original problem. Needless to say then, the role which simulating circuitry plays in the environment where analogies are used as the basic principals for the construction and functioning of new devices is even larger. Among the latter is the analog neural network created in the Munster Institute of Applied Physics (FRG) in 1989 by Dr. H. Purwins' group. Their network is the closest to what was proposed by the author in 1986. For this latter reason, I shall emphasize here the significance of the work. The most basic demand on any neural network is to have a significant number of discernable attractors.



The Lienard Oscillator





A plexus of coupled individual oscillators of Lienard type, each having several stable limit cycles (equilibrium might or might not be among them) is definitely a candidate for a dynamical system with a rich set of attractors. The network built by Purwins' group is a hardware realization of the excitable media concept in which the type of coupling between individual dissipative structural units defines the operative properties of the network. Nonlinear current-voltage characteristics synthesized by Purwins precluded the appearance of oscillatory regimes in individual "neurons," but even with this severe constraint, his network has demonstrated a magnificent abundance of experimentally observed distinctive attractors. The latter fact is the most important lesson for the present effort. Details on the network are in Purwins[+]. The present author's first (1986) multiple-limit cycle self-oscillatory "neuron" design is presented above and is self-explanatory. Apparently, it can serve as a potential generator of a set of attractors when coupled into a network.

II. Excitable Media At Hand

In deriving dynamical equations for self-oscillators, the source of energy (battery) is traditionally viewed only as a static, invariant parameter. The fact that the power source can be finite and therefore suffers time-dependant change of its basic characteristic while the self-oscillator is in its excitable state (internal resistance, for example) has, to the best of this author's knowledge, never entered the field of nonlinear oscillations or control theory. Meanwhile, the mitochondria participating in complicated vibrational motions within the

---

[+] Purwins, H., "Dissipative Pattern Formation in Experimental Analog Network," In H. Haken, ed., "Neural Computers" (Springer, 1990).

eukaryotic cell is definitely undergoing changes in its energy capacity. The fact is known and is particularly pronounced just before the beginning of mitosis, but mitosis is absent in neural cells. This makes a strong motivation for a phenomenological description of structural self-changes in excitable media. When can these changes lead to an even remotely analogous phenomena in an oscillatory, excited media? When can they not? Below is presented a formal scheme which allows us to pose that question in a framework of dissipative structures. Only structural units of this scheme are fully investigated[++]. The network per se has not yet been built. The mathematical premise drastically depends on the types of coupling between the oscillators and is open for flexible variations. What follows next will be a formal interpretation of these so far vaguely defined, unspecified terms, questions, and intentions. We begin from the oscillator

$$\ddot{X} + \mu \cdot F(X, \lambda) \cdot \dot{X} + X = 0$$

whose qualitative behaviour is fully understood and well investigated. For all admissible values of parameters μ, λ the oscillator is assumed to be a dissipative dynamical system. Recall that dynamical systems are dissipative if

$$\left([\dot{x}(t)]^2 + [X(t)]^2 \right) \in [0, R)$$

for sufficiently large $t > 0$ and arbitrary $\dot{x}(0)$ and $x(0)$; $R > 0$ is a fixed number. Let $S(\lambda, \mu)$ be a finite number of stable limit cycles. In quasilinear frameworks (small μ) there exist $F(X, \lambda)$ such that there is only stable equilibrium (0,0) at $\lambda = 0$; stable equilibrium and one stable limit cycle at $\lambda = \lambda_1$ and so on, in the case of generic bifurcations

$$S(\lambda, \mu) = \{SE, (SC)_1 \ldots (SC)_j\} \quad j = 1, 2 \ldots N.$$

where SE is a notation for stable equilibrium, and $(SC)_K$ means stable limit cycles existing for $\lambda_K < \lambda < \lambda_{K+1}$ where λ_K are bifurcation values of λ. Consider

$$\frac{\partial c}{\partial t} = D \cdot \frac{\partial^2 c}{\partial x^2} + \varphi\left(x, c, \frac{\partial c}{\partial x}, |u|\right)$$

$$\ddot{u} + \mu \cdot f(u, c) \cdot \dot{u} + u = \int_{\alpha(x)}^{\beta(x)} R(x, s) \cdot u(t, s)\, ds$$

u(t,x) and c(t,x) are unknown functions. This scheme includes all "inhibitor-stimulator" pattern formation paradigms so far considered in one-dimensional settings, but it contains the features which were never introduced before. In this formulation a, β, φ, R are left unspecified and that gives significant flexibility for variations in model building. Purwins' network corresponds to a particular choice for the above functions. The lack of space precludes the author from describing the qualitative dynamics of this continuum model in concrete terms here. It will be done somewhere else through further publications. One thing, however, should be stressed at once. The change in values of λ causes what was mentioned as structural change in the basic oscillatory unit. With λ (or C) undergoing transition through bifurcational value λ_K changes the type of oscillatory unit, since it changes the number of levels of excitations available.

[++]Saet, Y.A., "Quasilinear and Relaxational Realms in Multiple Regimes Self-oscillators," International Journal of Non-linear Mechanics, (Nov., 1991).

III. On Simulating Annealing Methodology

Physics of condensed matter generated a class of specific problems relating to finding minima of functions of large numbers of independent variables. Thus the ideology of statistical physics and mechanics was transferred to the fields of optimization and algorithm design. As for the neural network, the simulated annealing procedure recently became important in training and speciation of the neural networks. The present author used the principle features of simulating annealing in a situation pertinent to limit-cycles of oscillators of the type which was previously described as the structural unit of oscillatory networks. The first experiments reported[+++] showed clearly a possibility to discern all existing limit-cycles at once. Which, in a sense, is equivalent to the parallel processing of a number of localized problems. Here I shall demonstrate the spirit of this approach in a manner which is equally applicable to limit-cycles or to equilibria. The result presented here is new and allows for the simultaneous location of a large number of extrema of smooth function with an arbitrary number of independent variables. The latter problem is of the type to which many techniques used in neural networks are now reduced. Let $V(x_1, x_2 \ldots x_N)$ be a smooth function of N variables. Consider the following system of stochastic differential equations:

$$\ddot{X}_i + \alpha \cdot \dot{X}_i + \frac{\partial V}{\partial x_i} = T \cdot \xi_i(t) \quad i = 1, 2 \ldots N$$

where $\alpha > 0$, $T > 0$, and $\xi_i(t)$ are uncorrelated white noises of unit intensity. The present author has proven the following statement. The microcanonical Gibbs distribution $\rho = \bar{e}^{\frac{1}{T}H}$ is a normalized solution of the stationary Fokker-Plank equation corresponding to the above system with $H = \frac{1}{2}\sum(\dot{x}_i)^2 + V(x_1, x_2 \ldots x_N)$, $\bar{e}$ being the normalizing constant. Let $T \to 0$, then one can immediately see that the density of probabilistic measure is unlimitedly growing at the point of absolute minimum. In fact, the whole probabilistic measure will be concentrated in the neighborhood of the minimum point. The above described preliminary experiments with limit-cycles imply that there is a band of values $T \to 0$ in which one can see in sequence not only the absolute minimum, but also most of the intermediate or local minima of the function V. The experiments with minima, however, were not fulfilled. In the case of "degeneration," there are several potential wells of equal depth. The author has found the second order criteria for the rate of concentration of probabilistic measure in the neighborhoods of these minima. The criteria involve Gauss-Kronecker curvatures of N-surface in (N+1)-space: $U = V(x_1 \ldots x_N)$ at the minima and allow for observation of all of them at once in the process of $T \to 0$, if the process is realized in hardware or numerical experiments.

[+++] Saet, Y.A., and G.L. Viviani, "The Stochastic Process of Transitions between Limit Cycles for a Special Class of Self-Oscillators under Random Perturbations", IEEE Transactions on Circuits and Systems (vol. CAS-34, No. 6, June 1987).

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

Design and Implementation of the ACU: An Expandable ANN Building Block

R.J. Schalkoff, K.F. Poole, R. Singh, R.E. Owens, J.N. Gowdy and A.E. Turner

Department of Electrical and Computer Engineering
Clemson University
Clemson, SC 29634-0915
email: rjschal@clemson.edu

ABSTRACT

The design and implementation of a space and time efficient, modular (extendable) and versatile Artificial Neural Network (ANN) building block is presented. This effort combines the expertise of neural network, VLSI design, and materials researchers. The resulting ACU module implements a 20-input, 20 output building block employing analog and digital circuitry and takes advantage of InP technology strengths by using high speed digital multipliers and low leakage devices for the analog current conversion. The current status of the effort and future research directions are shown.

INTRODUCTION

Artificial Neural Networks (ANNs) offer alternatives to conventional solutions in a myriad of application areas. This includes problems in pattern recognition, including speech and image processing [1]. A plethora of ANN architectures exist, including: (i) the popular feedforward structure; (ii) recurrent structures typified by Hopfield networks; and (iii) other structures such as Kohonen's self-organizing net and ART1 and ART2. A given application may dictate preference of one structure over another.

Much of the current emphasis on ANN design and application centers around simulation of the networks. The next phase of ANN technology is likely to concern the hardware implementation of ANN-based solutions. To this end, research at Clemson University has emphasized the implementation of large-scale ANNs, where network size (in terms of weights or neurons) becomes a considerable challenge to VLSI technology. An excellent summary is found in [2-3]. The scaling of an ANN is especially significant in imaging applications, where the volume of image data leads to large ANNs [4].

Rather than concentrate on a specific ANN architecture, the effort involved identification of the common computational characteristics of layered and recurrent networks and the expansion or scaling of this computation to networks of arbitrary size. The solution is strongly dependent on the ability to compute inner products and then transfer these results to other units which do the same. Training of the neural units is assumed to occur off-chip. Other topics considered in the design phase include: weight accuracy (quantization) [5], arithmetic accuracy, speed-area tradeoffs, and analog vs. digital implementation. Some design issues are explored below.

DIGITAL VERSUS ANALOG APPROACH

An examination of the literature shows that analog circuits [6] were preferred in the initial attempts to implement an ANN integrated circuit. More recent work [7-8] describes the use of digital circuits in this application. Both approaches have their strengths and weaknesses [9] but, as ANN's are in an evolutionary phase, we believe that the digital/analog hybrid approach will advance the ANN field compared to systems based on one technology alone.

Hybrid technologies have been used by other researchers [10] and the approach used in this work extends this to capitalize on fundamental technology advances. In the long run, as the ANN field matures, the practical difficulties associated with a technology (not the inherent advantages or disadvantages) may force the use of a particular solution.

TECHNOLOGY RELATED ISSUES AND POSSIBLE SOLUTION

As discussed earlier, this hybrid technology approach has the potential to yield an ANN which is useful in large scale applications. Digital multipliers based on conventional Si technology (CMOS, bipolar and Bi-CMOS) occupy too large a chip area to allow large numbers to be placed on a chip [9]. Out of the various semiconductor transistors currently under consideration, the resonant tunneling transistor (RTT) offers the highest speed and minimum area that can be achieved in any three-terminal switching device. Therefore, the limitation of Si technology can be circumvented by using InP based ultra high speed vertical resonant tunneling transistors in the design of a super-fast and low area multiplier. Due to the higher saturation velocity and the coefficient of impact ionization at a higher electric field, higher thermal conductivity, better surface passivation properties and superior radiation resistance properties, InP transistors have exhibited superior performance over their GaAs counterparts [11].

In order to take advantage of a number of desirable features of Si substrates (availability in large area, high thermal conductivity compared to III-V semiconductor materials, higher mechanical strength, and low-cost) we propose to use an InP/buffer layer/Si substrate in the design of the digital I/O blocks, the multiplier and the summation circuitry. The analog (sigmoid) block with a variable shape will use conventional silicon circuitry.

ACU MODULES

An ACU module is shown in Figure 1. Currently, the ACU design provides a 20 input to 20 output neural mapping capability. In order to have a useful building block, connections to another layer and layer expandability are necessary. Note that the computation of each unit's net activation has been separated from the activation - output mapping (i.e., 'squashing'). This allows: (i) selection of different families of squashing functions, but more importantly, (ii) modular expansion of ACUs to large-scale designs.

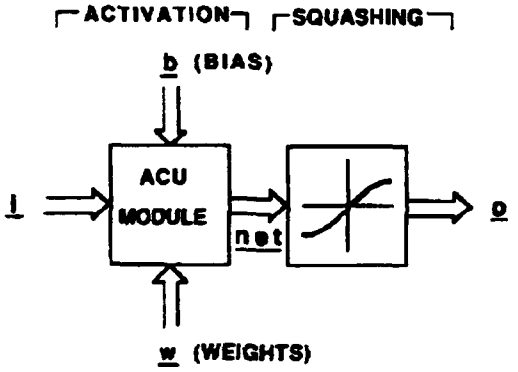


Figure 1. Overall ACU-element Based Structure

ACU DESCRIPTION AND SYSTEM PERFORMANCE

A sketch of the ACU layout and system block diagram are shown in Figures 2 and 3. Each ACU has 20 inputs and the chip architecture was designed to reduce interconnection busses and hence chip area. The estimated area for each multiplier is 0.1 mm^2 and the minimum time delay is 100 ps. Using a multiplier per input increases the parallelism and redundancy in the system but increases the chip area. Reduction in chip area is achieved by using

analog summing and squashing circuitry. The ACU design is complicated by the introduction of InP technology, but the estimated gains in performance and traditional areas of compromise are expected to outweigh the increase in fabrication complexity [4].

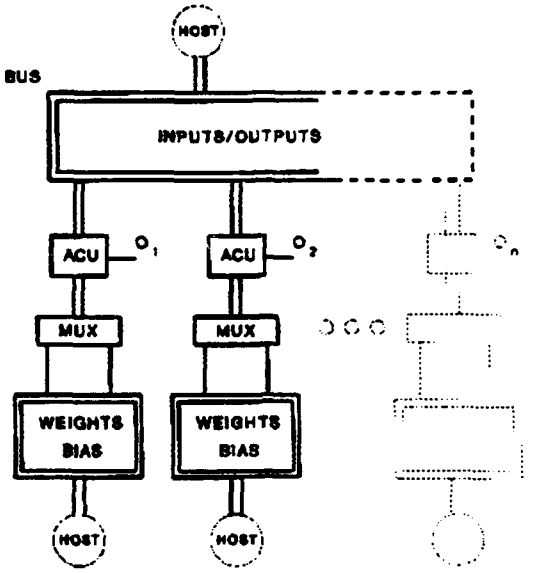


Figure 2. System Block Diagram

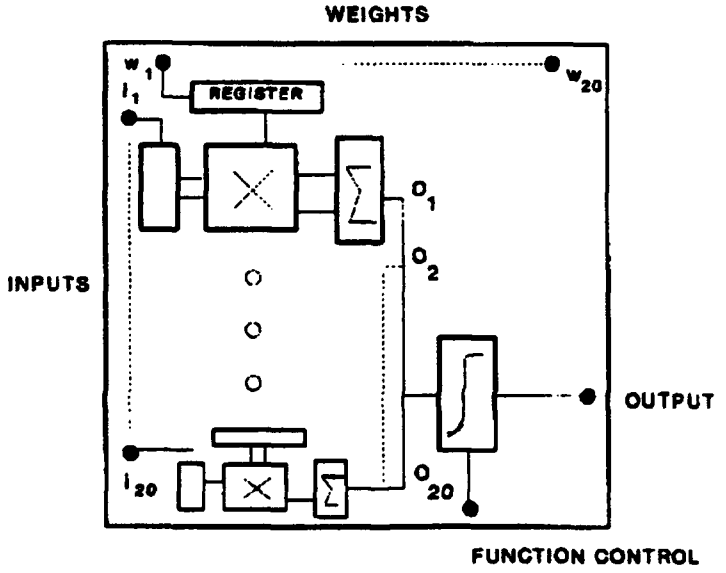


Figure 3. ACU Chip

Weight accuracy is 8 bit and multiplexing of the 400 unit weights is necessary to reduce the package pin count. Host computers are used for loading weights and inputs on separate busses. This provides the ability to alter weights dynamically. System performance is improved by having calculations proceed during the loading of the weights. The analog variable shape (sigmoid) function improves the accuracy of the threshold function and at high current levels, noise is not expected to be a problem.

The performance will eventually become I/O bound as the number of ACUs increases to cope with increasing inputs and weights in the expandability mode. Speed will be limited by the ability of the host machines to service the ACUs rather than the speed of the InP based ACU. The chip layout design is pad limited and the ACU is expected to occupy an area of 8 x 10 mm^2.

EXPANSION OF THE ACU MODULE

Expansion of ACU modules for large scale problems is based upon examining the input-net activation computation of a q-input, q-output[1] unit:

$$\underline{o} = W \underline{i} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} \underline{i}_1 \\ \underline{i}_2 \end{bmatrix} \tag{1}$$

Any decomposition of (1) yields a candidate architecture for ACU implementation, however, some decompositions are preferable. For example, one grouping of the computation in (1) yields:

$$\begin{bmatrix} \underline{o}_1 \\ \underline{o}_2 \end{bmatrix} = \begin{bmatrix} W_{11} & 0 \\ 0 & W_{22} \end{bmatrix} \begin{bmatrix} \underline{i}_1 \\ \underline{i}_2 \end{bmatrix} + \begin{bmatrix} 0 & W_{12} \\ W_{21} & 0 \end{bmatrix} \begin{bmatrix} \underline{i}_1 \\ \underline{i}_2 \end{bmatrix} = W_d \begin{bmatrix} \underline{i}_1 \\ \underline{i}_2 \end{bmatrix} + W_{ad} \begin{bmatrix} \underline{i}_1 \\ \underline{i}_2 \end{bmatrix} \tag{2}$$

Notice in the W_d-based activation computation, only units involving $\underline{i}_1$ connect to $\underline{o}_1$ and only units involving $\underline{i}_2$ connect to $\underline{o}_2$. Similarly, in the W_{ad}-based activation computation, units involving $\underline{i}_2$ only connect to $\underline{o}_1$ and those involving $\underline{i}_1$ connect to $\underline{o}_2$. Thus, a decomposible, local interconnect strategy is possible.

Finally, from (2), the overall activation output is found from

$$\underline{o} = \underline{o}_d + \underline{o}_{ad} \tag{3}$$

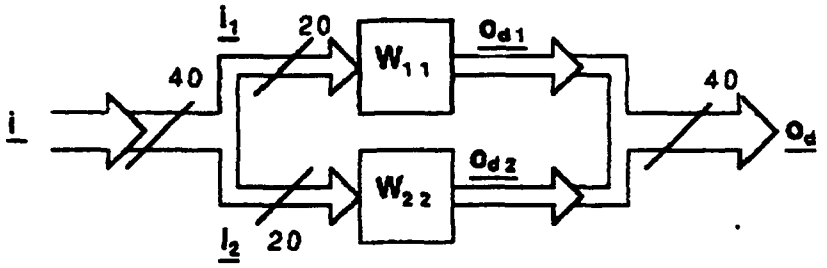
where the 'diagonal' ($\underline{o}_d$) and 'anti-diagonal' ($\underline{o}_{ad}$) outputs correspond to the respective W_d and W_{ad} products in (2). This process is shown in Figure 4.

SUMMARY/CONCLUSIONS

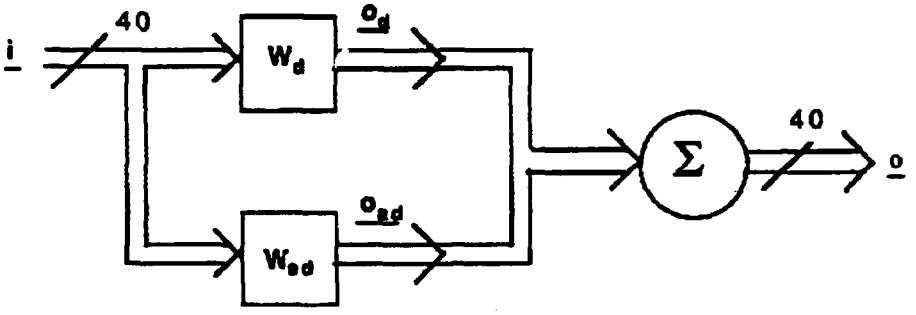
The ACU architecture and InP technology offers unique solutions to the well known bounds of artificial neural network chips which result in tradeoffs between interconnects, speed, chip area, and number of pins. The solution provides the expandability and flexibility necessary to achieve the large networks which are needed for many interesting applications.

---

[1]The number of inputs does not necessarily equal the number of outputs. We simply use this case for illustration.



(a) Decoupled ACUs



(b) Composite Net Activation

Figure 4. ACU Expansion

REFERENCES

[1] R.J. Schalkoff, Pattern Recognition: Statistical, Structural and Neural Approaches, John Wiley and Sons, 1992.

[2] Treleaven et al., "VLSI Architectures for Neural Networks," IEEE Micro, December 1989, pp. 8-27.

[3] Goser et al., "VLSI Technologies for Artificial Neural Networks," IEEE Micro, December 1989, pp. 28-44.

[4] DARPA, 1988 Neural Network Study, AFCEA Int. Press, Fairfax, VA, 1988.

[5] Hollis et al., "The Effects of Precision Constraints in a Backpropagation Learning Network," Neural Computation, Vol. 2, pp. 363-373.

[6] C. Mead, Analog VLSI and Neural Systems, Addison-Wesley, New York, NY, 1988.

[7] A. Maski, Y. Hiyai, and M. Yameda, "Neural Networks in CMOS: A case study by Akira Masaki, Yuzo Hiuai, and M. Yanuda," IEEE Circuits and Devices Magazine, Vol. 6, No. 4, pp. 12-17, 1990.

[8] U. Ramacher, J. Beichter, and N. Bruls, "Architecture of a General-Purpose Neural Signal Processor," ICJNN Vol. 1, Silicon Implementations of Neural Networks, IEE Proc. F., Vol. 138 No. 1, pp. 3-11, 1991.

[9] B.E. Boser, E. Sackinger, J. Bromley, Y. LeCun, R.E. Howard and L.D. Jackel," An Analog Neural Network Processor and its Application to High-Speed Character Recognition," ICJNN Vol. 1, pp. 415-420, 1991.

[10] H.P. Graf, R. Janow, C.R. Nohl, and J. Ben, "A Neural-Net Board System for Machine Vision Applications," ICJNN Vol. 1, pp. 481-486, 1991.

[11] R. Singh, M.J. Semnani, J.R. Cruz, and S. Sinha, Proc. Second International Conference on Indium Phosphide and Related Material, IEEE Catalog #90CH 2859-7, p. 340, 1990.

[12] Intel Corporation, 80170NX Technical Reference.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

Dynamically Reconfigureable Neural Network for Landmark Recognition

Ren C. Luo†, Harsh Potlapalli† and David W. Hislop*

†Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911
*US Army Research Office
PO Box 12211
Research Triangle Park, NC 27709-2211

Abstract

In this paper we present a new update rule for self organizing neural networks that increases learning stability. This rule, based on update normalization, prevents overshoot of the update. Also, we present a novel configuration approach which makes the network adaptable to new patterns with minimal delays.

Introduction

Autonomous mobile robots rely on sensors for guidance information to locate and identify landmarks when traversing their environments. Common examples of landmarks are mile-markers and street signs. Mile markers provide information about the global position of the robot while street signs provide local information. Some of the problems associated with landmark recognition are that the size of the object is subject to change depending upon the imaging distance. Also, if the viewing angle is not head-on, the aspect ratio may also change. In a dynamic environment the number and types of landmarks entering the scene is constantly changing and the recognition algorithm must be able to identify these new landmarks.

In this regard, visual sensors, such as intensity images,can convey more scene information than most other sensors. Hence, vision based sensing is the information collecting strategy of choice for most mobile robot guidance applications. Since the robot must make quick decisions based on the information contained in the landmarks, the recognition algorithm must be able to satisfy the recognition speed requirements for smooth motion. Also, the algorithm must be able to smoothly integrate into its object database for any variants of known objects or new objects that may appear in the scene that may be of use for navigation. In this paper, we present a Kohonen network architecture to meet these constraints.

First we present a brief list of current landmark recognition and mobile robot guidance techniques. Next, we justify the choice of the neural network model. We also present our results with these model and lastly, our conclusions on the effectiveness of this strategy.

Current Landmark Recognition Strategies

The objective of the landmark recogntion algorithm is to extract landmarks as quickly as possible. The algorithm should be able to overcome scale changes in the landmarks which may occur due to changing imaging distances. To overcome this problem, landmarks are often designed to stand out from the rest of the image in terms of their color or shape. In some instances, the landmark is placed on a retro-reflective surface which is illuminated periodically by a strobe light from the vision system[1]. Due to

[1] T. Takeda, A. Kato, T. Suzuki and M. Hosoi. Automated vehicle guidance using spotmark. In ICRA, pages 1349-1353, 1986
W.D. Holcombe, S.L. Dickerson, J.W. Larsen and R.A. Bohlander. Advances in guidance systems for industrial automated guided vehicles. In SPIE Mobile Robots III, vol. 1007, pages 288-297, 1988.
R. Ringling. Landmark tracking and automatic locating of vehicles for material handling. In Material Handling focus, 1988

the retro-reflective background and the intense beam of the strobe light, only the landmark is visible in the image. The extraction process is then simply reduced to locating the one bright region in the image. This approach requires that all the landmarks be identical. Guidance is provided by keeping a count of the landmarks detected. Thus, if any landmark is missed, the guidance strategy fails. Also, all the other information in the scene cannot be captured within the same image. Other limitations of this approach are obvious.

Landmark detection without the aid of special lighting conditions has been studied by Fukui[2]. The algorithm scans every column of the thresholded intensity image to construct a probabilistic model of the dark and bright patterns which is used to predict the landmark. Such a task presents considerable computational burden especially if the image contains a large number of objects, only a few of which are relevant for the guidance.

Another approach has been to use a laser beam mounted on the robot to scan predetermined locations for bar codes which can be interpreted for location information[3]. In some other instances, laser transceivers on the robot received the beam bounced off of corner cubes placed along the edge of the path to correct the trajectory of the robot[4]. In this approach, as in the retro-reflective approach, there is no provision for error-correction. While the path of the robot can be maintained accurately, it is not possible to predict the exact location of the robot with the guidance system alone.

A major disadvantage of these image processing oriented landmark detection algorithms is that each requires substantial processing of the image before any information regarding the landmark can be generated. This processing that is typically in the form of clustering, segmentation and identification has a computation overhead that may be difficult to achieve in mobile robot applications where speed of recognition is essential. For example, locating a street sign may be accomplished with small investments of computation; however, interpreting the meaning of the sign may require a lot of effort.

On the other hand, neural networks generally have a lower computation requirement once they are trained up. Once the possible landmark candidate has been located, the identification can be performed within the time it takes to percolate the landmark subimage through the network. It is true that the training of neural networks requires considerable computation which may be of the same order as that required by the image processing oriented techniques. However, a more pertinent criterion is the time required to identify a given pattern; in this regard, the neural network can be much faster. Also, the basic approach does not change even if the patterns to be detected change.

There are a number of neural network models that have been used for pattern recognition applications. A broad classification of such models can be represented as Back propagation[5], Adaptive Resonance Theory (ART)[6] and Self Organization[7]. Of these, back propagation models are useful for applications where severe distortion in the input maybe expected, such as hand written character applications. Backpropagation models are unable to recognize patterns from any class that was not represented in the training set without retraining the network. ART deals with an "oscillatory" weights model in which the network adjusts the weights for every input till a known "state" is reached or the the pattern is learned as a new class. Kohonen models are similar to vector quantization where the space of input patterns is scattered with neurons which are iteratively adjusted till every class has at least one neuron. For the mobile robot navigation application studied in this paper, we need a network that has the fastest integration time for new patterns. Also, the training time and the weight storage overhead must be low. In the next section we describe a Kohonen model with new update rules that incorporate learning stability. We present learning curves that compare the performance of our method with the standard Kohonen learning when the networks were trained with intensity images of actual street signs.

Self Organizing Neural Networks

The training algorithm in self organization attempts to find an optimal cover of the input space by the neurons. Training is concluded when the neuron assigned to each class is at an average minimum distance to all presented patterns of that class. This presents the one major difficulty with these types of networks: convergence is not guaranteed.

---

[2] I. Fukui. TV image processing to determine the position of a robot vehicle. Pattern Recognition, vol. 14, pages 101-109, 1981

[3] R. Hart. A laser guided AGV application at Ford-Indianapolis. In International Material Handling focus, 1988

[4] T.Tsumura, M. Hashimoto and N. Fujiwara. A vehicle position and heading measuring sytem using corner cube and laser beam. In Position, Location and Navigation symposium, pages 47-53, 1988

K.Nishide, M.Hanawa and T. Kondo. Automatic position finding vehicle by means of laser. In ICRA, pages 1343-1347, 1986

[5] A.E. Bryson and Y.C. Ho. Applied Optimal Control. Blaisdell,1969

J.L. Rumelhart and J. McClelland. Parallel Distributed Processing. MIT Press 1986

Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, vol. 1 pages 541-551, 1989

K.Fukushima. Neural network model for selective attention in visual pattern recognition and associative recall. Applied Optics, vol 26, pages 4985-4992,1987.

[6] G.A. Carpenter and S. Grossberg. Self organizing of stable category recognition codes for analog input patterns. Applied Optics, vol 26, pages 4919-4930

G.A. Carpenter and S. Grossberg. A massively parallel architecture for a self organizing neural pattern machine. Computer Vision, Graphics and Image Processing, vol 37, pages 54-115, 1987.

[7] T. Kohonen. Self organized formation of topologically correct feature maps. Bilogical Cybernetics, vol 43, pages 59-69, 1982

T. Kohonen. *Self Organization and Associative Memory*. Springer Verlag, 1988.

Unlike supervised learning strategies where convergence can be proved if a small learning rate and a small starting set of weights is used, it is difficult to predict the outcome of the training exercise with Kohonen networks. To see this, we can visualize each neuron as the center of an N-dimensional hypersphere that contains all examples of the particular class. Initially the neurons start out at random positions in the input space and during training, the "winning" neuron for each class is slowly moved to the center of the covering sphere. Now, if the variance of each class is high, then as the sphere moves to cover some members of the class, some others will be left out. Also, if the inter-class variance is too low, the spheres may intersect. Either of these cases is pathological and leads to oscillations and instability. Kohonen suggested that by normalizing all vectors, inputs and neurons, to unit length, this problem could be avoided. In this case, all the vectors that belong to the same class lie within a circle, the center of which would be the optimal location of the neuron that represents the class. This may lead to the conclusion that the best training strategy would be one that computes the center of this circle from the given data points. However, this is computationally intensive task especially with vectors with high dimensions. Also, there are geometrical constraints; for example, the data might be from just one part of the actual circle; a circle is uniquely defined by 3 points and with raw data it is hard to decide which three of the data could be used.

A simpler solution to the optimal neuron location problem is to start with random locations of the neurons that may be iteratively adapted to the input data so that after a small number of iterations, the neurons attain some pre-defined degree of optimality such as when all the training set, as well as all the test set, patterns are correctly classified. In the next section we present the learning rules for the Kohonen self-organizing map that overcome some of the disadvantages of the network.

Learning Rules for Self Organizing Networks

Unlike backpropagation, the representative neurons for each class are not predetermined. Also, each neuron is represented by a weight vector of the same dimension, M, as the input data. For every pattern presented to the set of neurons, the closest neuron is located, closeness being measured in terms of Euclidean distance. For this reason, both the input patterns and the weight vectors are normalized to unit length. To overcome the problem of oscillations in Kohonen networks we have developed the following learning rules based on update normalization. The closest neuron is updated as:

$$w_{ij}(n) = w_{ij}(n-1) + \frac{\alpha(n)(x_j - w_{ij}(n-1))}{(\|\mathbf{x} - \mathbf{w}_j\|) + 1}, j = 0, \ldots, M-1 \tag{1}$$

where, w_{ij} is the j^{th} element of the winning neuron i, α is the learning rate and x_j is the corresponding element in the input vector. Also, a small, monotonically decreasing, neighborhood of neurons around the winning neuron are also updated with respect to the input. The philosophy behind this approach being that neurons that are close together typically belong to classes that are also close together. We have found that a good choice for initial value of the neighborhood function is the square root of the maximum distance between any pair of neurons at the begining of the first iteration. This neighborhood update moves the neighborhood neurons in the general direction of their classes, thereby increasing convergence rates. However, this update is weighted by the distance between the input and the neuron. That is, if neuron k is not the winning neuron,

$$if \ \|\mathbf{w}_k - \mathbf{w}_i\| < N_c$$

$$w_{kj}(n) = w_{kj}(n-1) + \frac{\alpha'(n)[x_j - w_{kj}(n-1)]}{\|\mathbf{x} - \mathbf{w}_k\| + 1}, j = 0, \ldots, M-1, and \ \alpha' < \alpha \tag{2}$$

The distance weighting ensures that the neighborhood neurons do not overcome the winning neuron. Also, the distance weighting has an accelerating effect such that as the winning neuron gets closer to the class it is expected to represent, the faster it moves toward the class. The factor of 1 in the denominator is used in case the distance is zero. The neighborhood distance is reduced over time so that eventually the neighborhood encloses only one neuron. This occurs when the neuron has moved into the covering circle. Ideally training is terminated when the neuron is at the center of the covering circle.

One of the shortcomings of self organizing neural networks is that if the variance within the input data set is low, one neuron may move much closer to the data than any of the other neurons. Then the network will consistently have the same output for all the inputs, i.e., the network has failed to learn the patterns. (On the other hand, backpropagation models will learn the patterns though it may take a very long time). In this paper we present a novel approach to tackle this problem. We flood the weight space with an excess of neurons, generally a multiple of the number of classes. This causes each neuron to concentrate on a very small part of the input space and prevents network saturation of the type discussed above. Also, these excess of neurons allow us the freedom of reassigning them to any new classes that may appear at a later time.

Further, in marked difference to the backpropagation training technique, we present each class sequentially so that neurons that are activated toward a class move far away from all other classes. This form of input presentation is not possible in backprop since it will cause the network to learn only this one pattern. We trained the self organizing model with images taken of street signs

Class	Threshold	New Classes						
		Light Ahead	No U Turn	No Left Turn	Ped. Cross	Rail Cross	Stop	Yield
Do Not Enter	0	-	-	-	-	-	-	1.4
	0	-	-	-	-	-	1.436	-
No Right Turn	0.2132	-	1.308	1.132	-	1.254	-	-
	0.134	-	-	-	-	-	-	-
Stop Ahead	0.9119	1.099	-	-	1.086	-	-	-

Table 1: Activation thresholds with 18 neurons in the weight space

such as *Stop* , *No Right Turn*, etc. The street signs were extracted using a segmentation routine. The minimum size of the sign was 20x20 pixels where the sign was just visible to the *human eye*. The maximum size used was 60x60 pixels where the camera was nearly alongside of the sign. To test the reconfigureability of the network we train the network with only 3 street signs with different numbers of neurons: 15, 18, 21, 24, 27 and 30. The ability of the network to learn the new signs is an important factor in the use of this model for mobile robot navigation. We use a Kohonen model in which the starting neurons are all initialized to small values and are normalized to unit length.

In each case we plot the average change in the distance between the input and the closest neuron. We also plot the change in the winning neuron due to the update. We expect both curves to be decreasing functions. One method to check for termination of training would be to check for zero change in the updated neurons. This means that further learning will fail to change the neuron any more. This is computationally expensive and also may overtrain the network and decrease its generalization capability. Hence, we continue to use the previous error metric: terminate training when all the test data and training data are recognized successfully. We present the learning curves for the first 100 iterations for each case in Figures(1,2,3). We notice that in the case of 15 input neurons the distance between the input and the winner is still quite high at the end of 100 iterations. Also, the classification error (the average number of inputs misclassified) is also high. Hence, we can conclude that 15 neurons are not sufficient to represent this input set with the desired error bounds.

From these figures it is also evident that for this application, 6 times as many neurons as there are classes are sufficient to learn the patterns. It is also evident that as the number of neurons is increased there is no significant increase in convergence rates. Also, there does not seem to be any greater improvement in the position of the winning neurons with respect to the class. The advantage of using more neurons is that provides the network greater flexibility to learn more new patterns.

The neurons that are not associated with any training set class are activated when new classes are to be introduced. At the termination of the training, we compute the maximum activation distance for each class. That is, for each neuron assigned to a class, we compute the distance to all the (training and test) members of this class. The maximum distance to the input for which the neuron is still the winner is the activation distance of the neuron. When a new pattern is presented, its distance to all the neurons is computed and the pattern is assigned to the closest class. However, if this distance is greater than the activation distance for that neuron, the new pattern is determined to belong to a new class. Then, the closest neuron that is not already pre-assigned to any class is updated to the position of the new pattern. Thus a new class can be incorporated into the weight space without any retraining overhead.

In Table 1, we show the activation thresholds of the neurons in each class computed from the training and test data. We present the results for the 18 neurons case. The table also shows the distance between new patterns and each existing class. We can see that the distance to the new pattern is greater than the threshold for the neuron. This means the neurons have sufficient discriminating ability. The closest of the remaining unassigned neurons is then moved to the location of the new input class. (A "-" in any cell of the table below indicates that the neuron in that row was not the closest to the input.)

Conclusions

In this paper, we have discussed some of the issues involved in landmark detection. We have presented a possible solution to the problem that can be dynamically reconfigured to incorporate new classes. We have also presented new update rules that can be used to increase learing efficiency. Finally, we have described some of the results obtained with these networks.

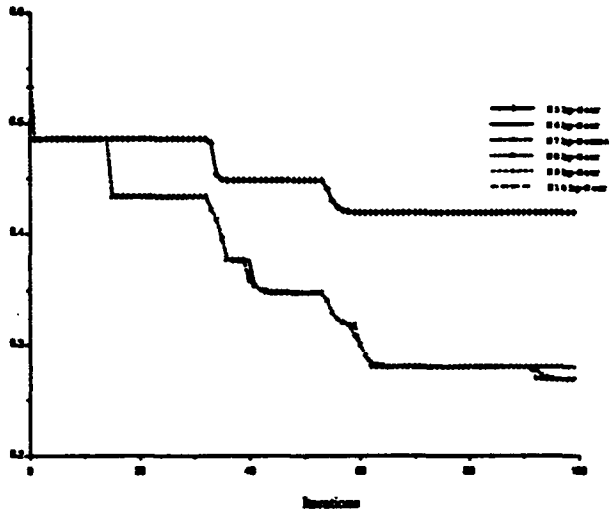


Figure 1: Distance between the winning neuron and the input vector

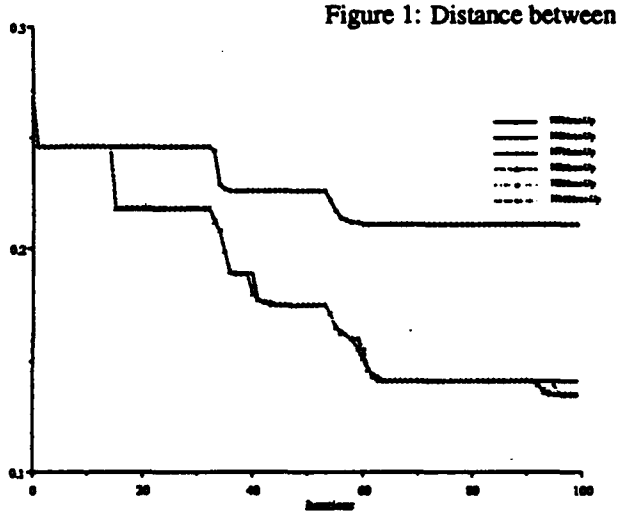


Figure 2: Update distance of the winning neuron

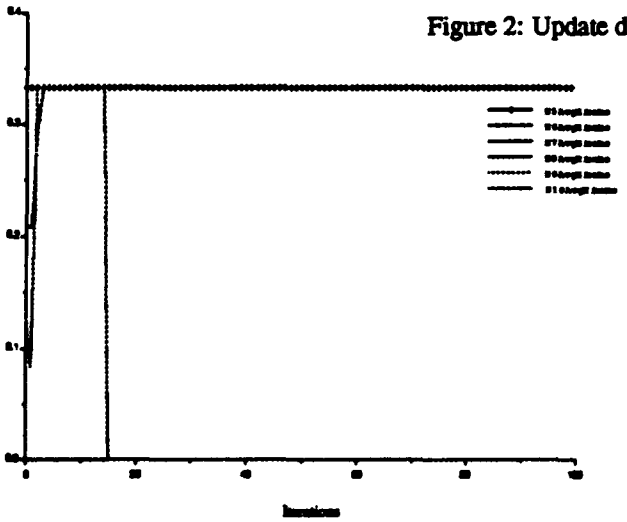


Figure 3: Average number of inputs misclassified

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop

24-26 August 1992 GACIAC PR 92-02

Adaptation & Learning in Control Systems: Application to Flight Control

July 1992

Walter L. Baker & Peter J. Millington
Autonomous Systems Group
The Charles Stark Draper Laboratory, Inc.
555 Technology Square, Cambridge, Massachusetts 02139-3563

ABSTRACT

A rapidly emerging area in the theory of automatic control involves the related concepts of *adaptation* and *learning*. According to one basic paradigm, an adaptive or learning controller can be viewed abstractly as a *variable mapping*, from desired and measured plant outputs to corresponding actuation signals, with adaptation and learning as the processes used to modify this mapping so as to improve future closed-loop system performance. More generally, adaptation and learning are perceived to be mechanisms for intelligent response in a system. A rudimentary investigation of this paradigm suggests that, in the context of flight control, many potential benefits may arise from a combination of adaptive and learning augmentation. These potential benefits include: facilitation of the control system design, development, and tuning process; compensation for insufficient or inaccurate *a priori* design information; improvement of operational efficiency; and enhancement of closed-loop system performance. In this brief paper, a hybrid adaptive / learning control methodology based on this perspective is motivated and described. The distinguishing characteristics of this approach and the benefits associated with its use are discussed. An example of a hybrid adaptive / learning control system applied to a nonlinear longitudinal aircraft model is also presented.

1 INTRODUCTION

An important, perhaps even defining, attribute of an intelligent control system is its ability to improve its performance in the future, based on past experiences with its environment. The concepts of *adaptation* and *learning* are often used to describe the processes by which this is achieved. In this paper we discuss control systems that are explicitly designed to incorporate and exploit both behaviors. Moreover, the use of connectionist systems as a representational framework is motivated and described in this context.

In a fundamental sense, the control design problem is to find an appropriate functional mapping, from desired and measured plant outputs to control actions, that will produce satisfactory behavior in the closed-loop system. In other words, the problem is to develop or evolve a function (a *control law*) that achieves certain performance objectives when applied to the open-loop system. The desired control law may be nonlinear and/or dynamic. In turn, the solution to this problem may naturally involve the synthesis of other mappings; e.g., from scheduling variables to the parameters of a local controller or local plant model, or from measured plant outputs to estimated plant states. The development of accurate mappings necessarily requires knowledge of the dynamical behavior of the *actual* plant. Such knowledge is often unavailable or expensive to obtain *a priori*. Thus, an *automatic* mechanism that could synthesize and maintain these mappings on-line would be an advantageous component of an advanced control system. The feedback required for on-line operation (i.e., the information needed to correctly generate the desired mapping) could be obtained through direct interactions with the plant (and its environment). In this way, adaptation and learning can be used to compensate for limited or inaccurate *a priori* design knowledge by exploiting information that is gained experientially. The advantages of adaptive and learning control systems derive mainly from their ability to accommodate (in varying degrees) poorly modeled, nonlinear, and time-varying dynamical behavior during on-line operation.

2 DISCUSSION

Adaptation and learning are *complementary* behaviors that can be used simultaneously (for purposes of automatic control) in a *synergistic* fashion (Baker & Farrell (1990)). The differences between the processes of adaptation and learning are important, but subtle. To some degree this is unavoidable, since there is no definitive dividing line. If, however, one considers the differences between *representative* adaptive and learning control systems, then several distinguishing qualities emerge. For example, learning control systems make extensive use of memory; adaptive control systems also rely on memory for their operation, but to a significantly lesser degree. A control system that treats every distinct operating situation as a novel one is limited to *adaptive* operation, whereas a system that correlates past experiences with past situations, and that can recall and exploit those past experiences, is capable of *learning*. The key differences, which are essentially a matter of degree, emphasis, and intended purpose, are discussed further in (Baker & Farrell (1992)).

In general, adaptation can be used to accommodate (slowly) time-varying dynamics and novel situations (e.g., those which have never before been experienced), but may be inefficient for problems involving significant unmodeled nonlinearity. Learning, in contrast, can be used to accommodate unmodeled nonlinear behavior, but may not be applicable to applications involving time-varying dynamics. In situations where the unmodeled dynamics are nonlinear, but time-invariant, the anticipatory character of learning systems allows for increased efficiency and improved performance to be achieved, relative to purely adaptive systems (due to the local nature and inherent lag in the adaptation process). In light of this, we are led to suggest that an intelligent control system might actually be comprised of three subsystems: an *a priori* compensator (which provides nominal performance), an adaptive component, and a learning component. Finally, as is discussed below, the successful employment of adaptation and learning for the purposes of automatic control implies that one has a feasible and effective means for their realization.

Connectionist Systems

Connectionist systems, including what are often called "artificial neural networks," have been suggested by many to be suitable structures for the implementation of adaptive and learning control systems. Common examples of connectionist systems include feedforward multilayer sigmoidal (Rumelhart, *et al.* (1986)) and radial basis function (Poggio & Girosi (1990)) networks. The popularity of such systems arises, in part, because they are amenable to simple gradient parameter adjustment algorithms, and can be implemented in parallel computational hardware. Perhaps more importantly, however, it is known that several classes of connectionist systems have the *universal approximation property*. This property implies that any continuous function can be approximated to a given degree of accuracy by a sufficiently large network (Funahashi (1989), Hornik, *et al.* (1989)). The universal approximation property is a key element of any application of connectionist systems to problems in automatic control.

A connectionist system can be implemented in three distinct forms: as a static feedforward, dynamic feedforward, or recurrent mapping. Static feedforward networks are used to realize *memoryless* nonlinear mappings. The structure of such networks can always be captured in an acyclic digraph. Dynamic feedforward networks are used to realize dynamical mappings having a *finite* impulse response, and can be represented using an acyclic digraph, with integration or delay operators at the output of all nodes. More general dynamical mappings, having an *infinite* impulse response, can be realized using recurrent networks; in this case, the network can always be represented using a *cyclic* digraph, with integration or delay operators at the output of all nodes. Note that *any* recurrent network representing a continuous or discrete-time dynamic mapping can be expressed as an equivalent *dynamical system* comprised of two *static* feedforward mappings separated by the appropriate operator. This fact suggests that standard techniques for adjusting static feedforward networks can be incorporated into algorithms suitable for recurrent networks.

To help clarify the discussion, assume that a connectionist system is to be used to describe the static mapping $\mathbf{y} = \mathbf{g}(\mathbf{u};\mathbf{p})$, where $\mathbf{p}$ is a vector of adjustable parameters. An algorithm is needed that will generate an appropriate $\mathbf{p}$ so as to achieve some desired objective. When the mathematical structure (in this case, the function $\mathbf{g}$) used to implement the mapping is continuously differentiable and the objective function J can be treated as a "cost" to be minimized, then gradient-based methods can be used. In the special case where the adjustable parameters $\mathbf{p}$ appear linearly in the gradient vector $\partial J/\partial \mathbf{p}$, the optimization can be treated using established linear methods (e.g., least squares). More generally, nonlinear optimization methods must be used.

In the event of a dynamic mapping $\mathbf{y}_{k+1} = \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p})$, where $\mathbf{x}$ is the internal state of the mapping, the problem can always be decomposed into two subproblems by representing the mapping in state-space form: $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p}_f)$, $\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k; \mathbf{p}_h)$. In this case, the two subproblems become that of estimating the parameters of the *static* mappings $\mathbf{f}$ and $\mathbf{h}$, and that of estimating the state $\mathbf{x}$ of the dynamical mapping (Livstone, *et al.* (1992)). Another approach to parameter adaptation in a recurrent network is to convert the recurrent mapping into an approximately equivalent dynamic feedforward mapping (i.e., by converting an infinite impulse response representation into one that has a finite impulse response) — this technique is closely related to the "backpropagating through time" strategy (Rumelhart, *et al.* (1986)).

Design & Implementation Issues

A number of technical issues must be addressed in any specific application of connectionist systems to adaptive and learning control. These issues include: controllability and observability; the effects of noise, disturbances, model-order errors, and other

uncertainties; convergence of connectionist network parameters, sufficiency of excitation, and nonstationarity; computational requirements, time-delays, and the effects of finite precision arithmetic. Most of these issues also arise in the application of traditional adaptive control systems.

Another design consideration involves the use of passive vs. active adaptation and learning strategies. *Passive* schemes are limited to whatever information happens to be available during the normal course of operation of the closed-loop system. In contrast, in an *active* strategy, the control system not only attempts to drive the outputs of the plant along a desired trajectory, but also explicitly seeks to improve the accuracy of the mapping(s) maintained by the control system. This is achieved by introducing "probing" signals that direct the plant into regions of its state-space where insufficient adaptation and/or learning has occurred.

An important factor to consider when employing connectionist systems for automatic control is the *environment* in which they will be used. Thus, for example, the quantity, quality, and content of the information that is likely to be available to the connectionist system during its operation critically impact its performance, and should be accounted for in the selection of a suitable approach. In our previous work, we have developed special-purpose connectionist systems that overcome some important limitations of popular connectionist architectures, when used in the context of automatic control (Baker & Farrell (1992)).

Incremental Learning

One nonlinear technique that is suitable for static mappings is the *gradient algorithm*: $\Delta \mathbf{p} = -\mathbf{W}(\partial J/\partial \mathbf{p})^T$, where $\mathbf{W}$ is a symmetric positive definite matrix that determines the "learning rate." More insight can be gained into the gradient learning algorithm through an application of the chain rule, which yields: $\Delta \mathbf{p} = -\mathbf{W}(\partial \mathbf{y}/\partial \mathbf{p})^T(\partial J/\partial \mathbf{y})^T$. The first partial derivative $\partial \mathbf{y}/\partial \mathbf{p}$ is the Jacobian of the outputs of the mapping with respect to the adjustable parameters, and is a known quantity. The second partial derivative $\partial J/\partial \mathbf{y}$ is the gradient of the objective function with respect to the mapping outputs, and may or may not be known exactly depending on the application.

A typical cost function is the quadratic form defined as the *sum*, over a finite set (or batch) of evaluation points, of the squared output error norms. If the goal is to have adaptation occur on-line, then an objective function of this form cannot readily be used. The main problem is that the set of possible inputs to the mapping will not consist of a finite set of discrete points. Consequently, it will not be easy to select a finite set of representative evaluation points, nor will it be feasible to guarantee that any or all of them are ever visited. Fortunately, various alternative objective functions that approximate this objective function are feasible and are often used in practice. By far, the most common objective function used for on-line learning in control applications is the *point-wise* function given by the square of the *current* output error norm.

This can be considered as a special case of the batch mode of operation, where the evaluation set contains a single (potentially different) point at each sampling instant. Learning algorithms that seek to minimize point-wise objective functions in lieu of objective functions defined over a continuum are referred to as *incremental* learning algorithms; they are related to a broad class of *stochastic approximation* methods. *Incremental gradient* learning algorithms operate by approximating the actual batch gradient with an *instantaneous* estimate based on the current evaluation point. The use of point-wise objective functions to approximate *batch* objective functions (i.e., those in which the evaluation set contains more than one point) *will generally not be successful unless special attention is given to the distribution of the evaluation points, the form of the learning algorithm, and the structure of the network.*

One well-known and widely used incremental gradient algorithm is the least-mean-square (LMS) algorithm (Widrow & Hoff (1960), Widrow, *et al.* (1976)). The LMS parameter update law is $\Delta \mathbf{p} = -\alpha(\partial J/\partial \mathbf{p})^T$, where the cost J is based on the point-wise squared error function and α is a scalar learning rate coefficient. Given certain assumptions (e.g., linearity, stationarity, ergodicity, Gaussian random variables, etc.), LMS can be shown to be convergent in the mean and mean-square, relative to the objective function that includes all evaluation points. Although the theory supporting the convergence of the LMS algorithm only applies to the special case of a linear network (among other assumptions), the basic strategy underlying LMS has been used to formulate a simple learning algorithm for nonlinear networks. This approach is the standard incremental gradient algorithm presently used by most practitioners for on-line parameter adjustment; it is equivalent to incremental "error back-propagation."

Spatially Localized Learning

Special constraints are placed on a connectionist system whenever learning is to occur on-line, during closed-loop operation; these constraints can impact the network architecture, parameter update algorithms, and training process. Assuming a passive update strategy is being employed, the experiential information (training examples) cannot be selected freely, since the plant states (and outputs) are constrained by the system dynamics, and the desired plant outputs are constrained by the specifications of the control problem. Under these conditions, the system may remain in small regions of its state-space for extended periods of time (e.g., near setpoints). In turn, this implies that the measurements used for incremental learning will remain in small regions of the input domain of the mapping being synthesized. Such "stasis" can cause undesirable side-effects in situations where parameter adjustments (based on incremental learning algorithms) have a non-local effect on the mapping maintained by the learning system. For example, if a parameter that has a non-local effect on the mapping is repeatedly adjusted to correct the mapping in a particular region of the input domain, this may cause the mapping in other regions to deteriorate and, thus, can effectively "erase"

learning that has previously taken place. Such undesirable behavior arises because the parameter adjustments dictated by an incremental learning algorithm are made on the basis of a single evaluation point, *without regard to the remainder of the mapping*. The idiosyncrasies associated with passive incremental learning under closed-loop control have precipitated the development and analysis of *spatially localized* learning systems (Baker & Farrell (1990, 1992), Millington (1991)).

3 EXAMPLE: PITCH RATE CONTROL

Modern flight control systems are used to provide closed-loop system stability and enhanced performance over a wide range of operating conditions. This can be difficult to achieve due to a number of circumstances including the complexity of both the vehicle and the performance objectives, and due to the presence of uncertainty in the design process. From a design standpoint, air vehicles can exhibit nonlinear and time-varying behavior (due to aerodynamic effects and fuel use), unmodeled dynamics (due to unmodeled aeroelastic effects), high dimensionality, multiple inputs and outputs, and the possibility of actuator, sensor, or other component failures. In this setting, there are potentially many applications of adaptive and learning control.

In the simple example presented here, the problem is to provide a command augmentation system (CAS) that will improve the longitudinal handling qualities (e.g., pointing capabilities) of a representative high performance aircraft. It is assumed that the available *a priori* model information is so limited that it is impossible or impractical to design in advance a control system that has fixed properties and also performs sufficiently well (thus, a fixed robust design cannot be used). Furthermore, it is also assumed that a model describing the desired closed-loop system behavior exists, and that this level of performance can be *realized* by the actual aircraft. Under these conditions, the objective is simply to track the output of the reference performance model. A more detailed description of a related experiment can be found in (Nistler (1992)).

Synopsis

The aircraft simulation used here is derived from a six-degree-of-freedom (6-DOF) F-15 "like" vehicle model, incorporating nonlinear aerodynamic effects (based on empirically derived tabular data), nonlinear engine dynamics, and nonlinear actuator dynamics (including rate and position limits). A more detailed description of the basic simulation can be found in (Brumbaugh (1990)). For the results reported below, the longitudinal and lateral / directional dynamics were decoupled, and a 3-DOF longitudinal vehicle model extracted. Neither sensor noise nor wind gust disturbances were modeled.

The performance model used for this experiment is essentially a gain scheduled fourth-order linearized model that describes the classical longitudinal dynamics of an aircraft. The model parameters are scheduled on altitude and Mach number over a representative flight envelope, and have been designed to meet basic military specifications for pilot handling qualities (i.e., MIL-F-8785C) at every trim condition within this envelope. Thus, for instance, the eigenvalues associated with the short-period and phugiod modes of the linearized model always satisfy the MIL-F-8785C requirements for Class IV (high performance) aircraft in Category A (maneuvering) flight. This performance model is designed to provide a specification of the desired transient response of an aircraft to pilot stick input commands. Further information regarding this particular model can be found in (Calvert (1991)).

Since only a single control (symmetric horizontal stabilator) was available in our 3-DOF simulation, we had essentially two choices: (i) to track a single output of the reference performance model or (ii) to minimize the norm of some multivariable tracking error. We choose to track a single output, namely the pitch rate of the performance model. Implicit in this approach is the assumption that accurate pitch rate tracking will yield acceptable responses in the other vehicle outputs. The actual feasibility of such an approach depends on the degree to which the reference pitch rate trajectory is realizable as well as the similarity between the dynamics of the performance model and the vehicle to be controlled.

All CAS controller designs were based on the following *a priori* model information only: a single fourth-order *linear* model derived by linearizing the nonlinear 3-DOF model at a trim condition of 20,000 ft and a Mach of number 0.65 (roughly the middle of the flight envelope). Thus, there were both unmodeled nonlinear effects and additional unmodeled dynamics (due to simulated actuator and engine dynamics). A model-based *linear* compensator was designed following a procedure similar to that described in (Anderson & Schmidt (1991)). An *adaptive* compensator was developed by combining and extending ideas presented in (Youcef-Toumi & Ito (1990) and (Anderson & Schmidt (1991)). Finally, a *hybrid* adaptive / learning control system was developed that combined the same adaptive compensator with a connectionist learning system (Baker & Millington (1991)).

A linear-Gaussian network (Baker & Farrell (1992)) employing 45 nodes and a spatially localized incremental gradient algorithm was used to capture the unmodeled nonlinear effects in the observed behavior of the vehicle. The hybrid controller was trained on 3,000 transient responses originating at randomly selected trim conditions throughout the flight envelope (roughly from sea-level to 40,000 ft, and Mach number 0.3 to 0.9). During training, both adaptation and learning were active in the hybrid system. Each transient response was generated by applying one of four predetermined stick commands to both the performance model and the CAS coupled to the nonlinear 3-DOF aircraft model. The four longitudinal stick inputs used were: a ramp, doublet, pulse, and sinusoid-sweep. All responses were made from an initial trim condition, with the throttle held constant at its trim value.

Results

Desired (reference performance) pitch rate response to a stick input doublet, and actual pitch rate tracking error, stabilator, and load factor responses are shown in Fig. 1 for each controller (linear, adaptive, and hybrid). Initially, the aircraft is in trim at an altitude of 35,000 ft and a Mach number of 0.68 (corresponding to a relatively low dynamic pressure condition). Note: this flight condition is different from the one where the *a priori* design model linearization was performed; thus, the design model does not even accurately represent the linearized behavior of the vehicle. These plots demonstrate a clear progression of improved tracking performance from the linear through the adaptive to the hybrid controller (the linear controller failed around 10 sec). The stick input used to generate these responses is shown in Fig. 2. A representative "slice" of the multidimensional mapping learned by the network is given in Fig. 3.

4 SUMMARY

Potential benefits associated with adaptive and learning control include: (i) facilitation of the control system design, development, and tuning process, (ii) compensation for insufficient or inaccurate *a priori* design information, (iii) improvement of operational efficiency, and (iv) enhancement of closed-loop system performance. Presently, sufficient scientific and engineering substantiation of these potential benefits is lacking. Nevertheless, on the strength of the preliminary results already obtained, as well as the basic theory that has been developed, further examination of the fundamental issues underlying adaptation and learning and their application to automatic flight control is warranted. In particular, new research and development efforts aimed at demonstrating key aspects of learning augmented control and estimation schemes are needed.

ACKNOWLEDGEMENT

This paper is based on work that was supported, in part, by: the Charles Stark Draper Laboratory, Inc., under IR&D Project No. 276; the Air Force Wright Laboratory, under Contract No. F33615-88-C-1740; the National Science Foundation, under Grant No. ECS-9014065; and the Naval Air Warfare Center, under Contract No. N62269-91-C-0033. Any expressed opinions, findings, conclusions, or recommendations are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

REFERENCES

Anderson, M. & Schmidt, D. (1991). "Error Dynamics and Perfect Model Following with Application to Flight Control," *J. of Guid., Control, & Dynamics*, vol. 14, no. 5.

Baker, W. & Farrell, J. (1990). "Connectionist Learning Systems for Control," *Proceedings, SPIE OE/Boston '90.*

Baker, W. & Farrell, J. (1992). "An Introduction to Connectionist Learning Control Systems," in White, D. & Sofge, D., eds., *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold.

Baker, W. & Millington, P. (1992). "Learning Augmented Flight Control for a High Performance Aircraft," in preparation, Draper Laboratory, Cambridge, MA.

Brumbaugh, R. (1991). "An Aircraft Model for the AIAA Controls Design Challenge," *Proceedings, 1991 AIAA Conference on Guidance, Navigation, and Control.*

Calvert, J. (1991). "Development of an Aircraft Simulation Performance Model for Flight Control System Gain Optimization Using Model Following Techniques," draft copy, Naval Air Development Center Report, Warminster, PA.

Funahashi, K. (1989). "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, pp. 183–192.

Hornik, K., Stinchcombe, M., & White, H. (1989). "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359–366.

Livstone, M., Farrell, J., & Baker, W. (1992). "A Computationally Efficient Algorithm for Training Recurrent Connectionist Networks," *Proceedings, 1992 American Control Conference.*

Millington, P. (1991). "Associative Reinforcement Learning for Optimal Control," Draper Laboratory Pub. T-1070, S.M. Thesis, Dept. of Aero. & Astro., MIT.

Nistler, N. (1992). "A Learning Enhanced Flight Control System for High Performance Aircraft," Draper Laboratory Pub. T-1127, S.M. Thesis, Dept. of Aero. & Astro., MIT.

Poggio, T. & Girosi, F. (1990). "Networks for Approximation and Learning," *Proceedings of the IEEE*, vol. 78, no. 9.

Rumelhart, D., Hinton, G., & Williams, R. (1986). "Learning Internal Representations by Error Propagation," in Rumelhart, D. & McClelland, J., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press / Bradford.

Widrow, B. & Hoff, M. (1960). "Adaptive Switching Circuits," *1960 WESCON Convention Record, Part IV.*

Widrow, B., McCool, J., Larimore, M., & Johnson, C. (1976). "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," *Proceedings of the IEEE*, vol. 64, no. 8.

Youcef-Toumi, K. & Ito, O. (1990). "A Time Delay Controller for Systems with Unknown Dynamics," *J. of Dynamic Sys., Meas., & Control*, vol. 112, no. 1.

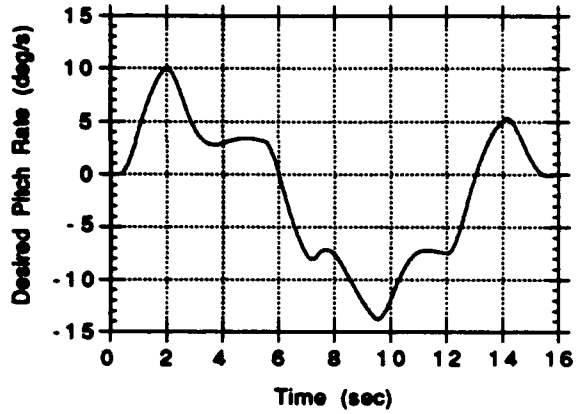


Fig. 1a. Desired pitch rate response.

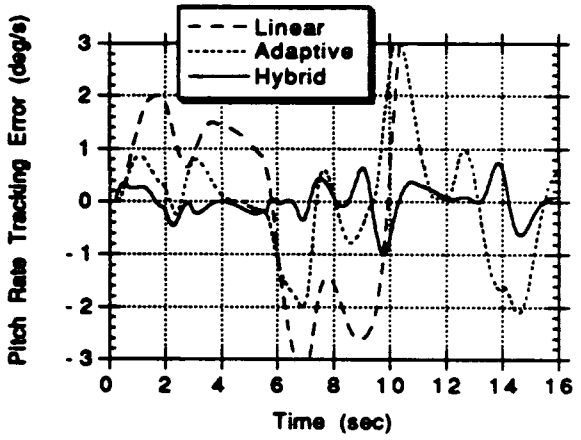


Fig. 1b. Pitch rate tracking error.

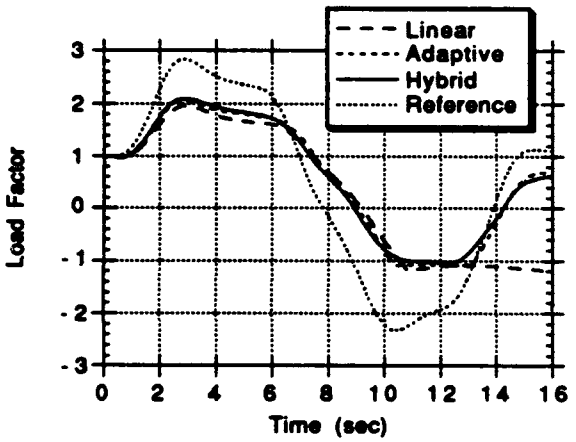


Fig. 1c. Load factor response.

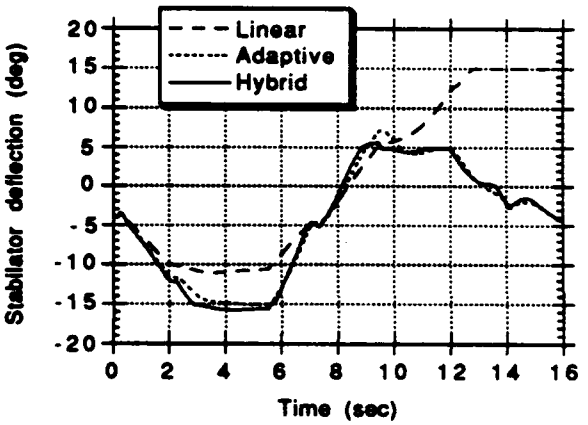


Fig. 1d. Stabilator response.

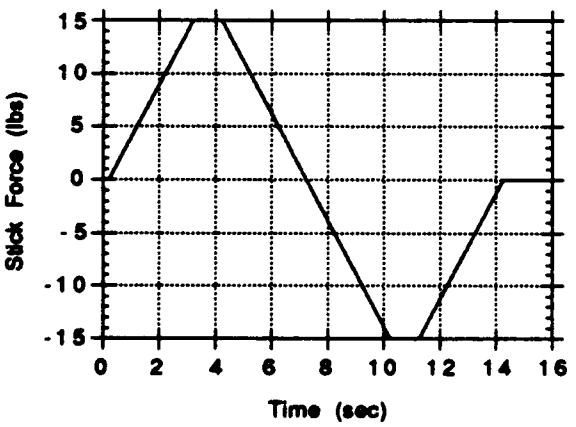


Fig. 2. Pitch doublet stick input used to generate transient responses in Fig. 1.

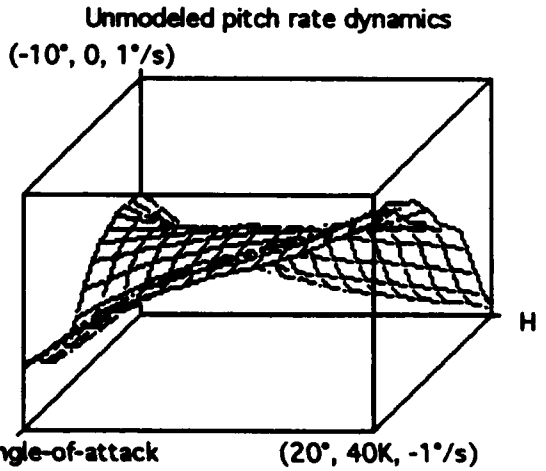


Fig. 3. A representative "slice" of the nonlinear mapping learned by the network.

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

A Low Cost Foveal Vision System[1]

R. Hecht-Nielsen and Y. T. Zhou

HNC, Inc.
5501 Oberlin Drive
San Diego, CA 92121

Abstract

A new foveated vision system is presented for target detection, classification, and tracking. This vision system measures features by foveal sampling – dynamically allocating visual resolution spatially and temporally to objects relevant to the task. Since foveal sampling uses sensor and processing resources more efficiently, it supports a decrease of several orders of magnitude in the amount of data processing needed to execute and complete a vision task. To reduce the cost and complexity of the required real-time sensor hardware, a spinning glass wheel has been used to build a foveated image sensor. A prototype demonstration model has been built to verify the concept of the foveated image sensor.

1 Introduction

One of the primary difference between the designs of the visual systems of animals and those of current machine vision systems is that animal sensors are foveated (i.e., they have higher resolution in the center of the field of view and lower resolution in the periphery), while human-made sensors have constant resolution throughout the field of view. Clearly, for image rendering applications such as television, constant resolution is ideal. However, for applications such as object acquisition, and object recognition, constant resolution image may be far from ideal, particularly as measured by system cost and real time performance.

In recent years, a few researchers in machine vision, neuroscience, and neural networks have argued convincingly that the best way to cut the costs and increase the capabilities of machine vision systems is to move to foveated sensors. In fact, a number of impressive demonstrations of the advantages and potential capabilities of such sensors have been carried out [2] [3]. Recently, we have developed a new vision system using a foveated sensor for target detection, classification, and tracking. To derive features, current machine vision systems start with a uniformly sampled, high-resolution image, and then carry out an enormous number of arithmetic calculations. In contrast, our vision system eliminates the need for the these expensive and slow intermediate steps. It measures features by foveal sampling – dynamically allocating visual resolution spatially and temporally to objects relevant to the task. Since foveal sampling uses sensor and processing resources more efficiently, it supports a decrease of several orders of magnitude in the amount of data processing needed to execute and complete a vision task. To reduce the cost and complexity of the required real-time sensor hardware, we have used a spinning glass wheel to build a foveated sensor.

2 Foveated Vision System

Our foveated vision system contains a Wheel of Fortune sensor (i.e., the foveated sensor), a saccade generation module, a detection module, a classification module, and a measurement and tracking module. Since

[1]This work was supported by DARPA under Contract DAAH01-92-C-R102 issued by the U.S. Army Missile Command.

[2]Y. Y. Zeevi and R. Ginosar. "Neural Computers for Foveating Vision Systems". In R. Eckmiller, editor, *Advanced Neural Computers*, pp. 323–330. Elsevier Science Publishers B.V., North-Holland, 1990.

[3]I. A. Rybak, N. A. Shevtsova, L. N. Podladchikova, and A. V. Golovan. "A Visual Cortex Domain Model and Its Use for Visual Information Processing". *Neural Networks*, vol. 4, pp. 3–13, 1991.

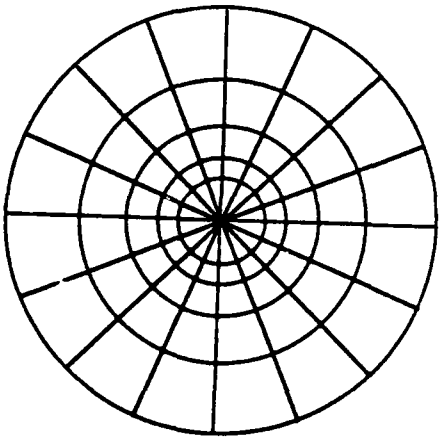


Figure 1. A foveal image feature sampling pattern rosette with 5 rings and 16 spokes.

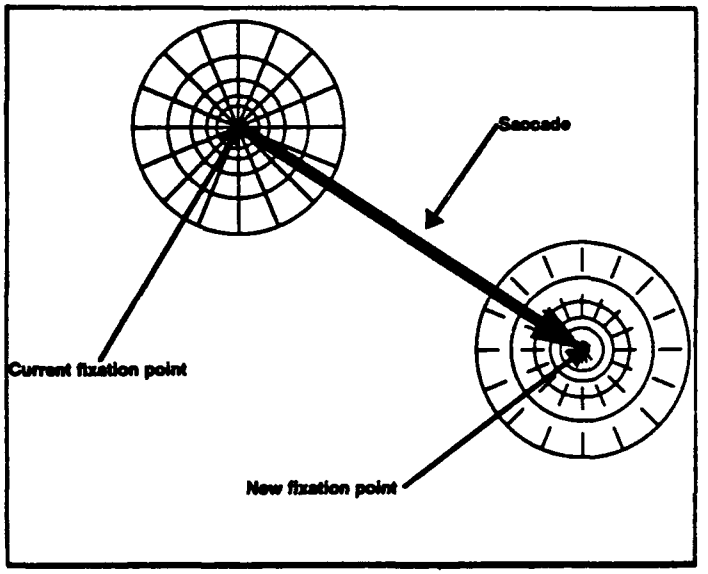


Figure 2. A saccade, the movement of the rosette from one fixation point to the next.

the system is designed to be easy update and maintain, each individual module can be updated without significantly affecting other components and new modules can also be added to the system.

The Wheel of Fortune sensor is a foveated image sensor. It is based on the modified version of Rybak's foveal rosette system [4]. The foveal rosette is a sampling frame of concentric rings and bisected by spokes as show in Figure 1. The radius of each ring is larger than the previous ring; this provides a nonuniform sampling pattern. At each intersection of a spoke and a ring, a set of features is gathered. The essential element of the rosette is the high density of the features near the center of the rosette and regular falloff of features to low density at the periphery of the rosette, not their regular spacing. The specific feature set is based on the concepts of Rybak[4]. The scale of the spatial frequency features depends on its sampling position in the rosette. The scale gets larger towards the center of the rosette to achieve a higher resolution. The point on the image that lies at the center of the rosette/foveal sampling pattern is called a fixation point. The movement of the rosette from one fixation point to the next, which is known as a saccade, is controlled by the saccade generation module.

The saccade generation module consists of a neural network and a set of fixed rules. The goal of saccade generation is to move the center of the rosette to a repeatable position on each object of interest within a scene as shown in Figure 2. The neural network is trained by the human "guiders" to move the rosette to fixation points leading to pre-selected nexus points on objects of interest. A nexus point defined as a fixation point that is located at a repeatably finable "center" of spatial frequency activity of an object. The network learns implicit methodologies for carrying out efficient scene searches for nexus points. Each saccade is less than one rosette radius in length. This ensures that the image information available to the saccade generator at a given time will be sufficient to derive the neural network. The fixed rules will be used whenever the probabilities of all the possible rosette moves by the network is low.

3 Wheel of Fortune Sensor Design

The Wheel of Fortune sensor concept is based on the technology used in the first television industry (c. 1928-1930). These television systems employed a large rotating wheel with a single spiral of pinholes near its periphery. A fixed aperture was located near top of the wheel. At the TV station the spinning wheel was designed such that at each moment, only one pinhole would be within the aperture (it would sweep across

[4] I. A. Rybak, N. A. Shevtsova, L. N. Podladchikova, and A. V. Golovan. "A Visual Cortex Domain Model and Its Use for Visual Information Processing". *Neural Networks*, vol. 4, pp. 3–13, 1991.

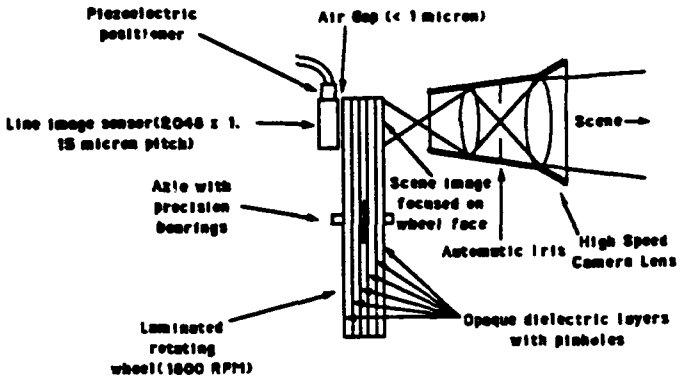


Figure 3. A spinning glass wheel.

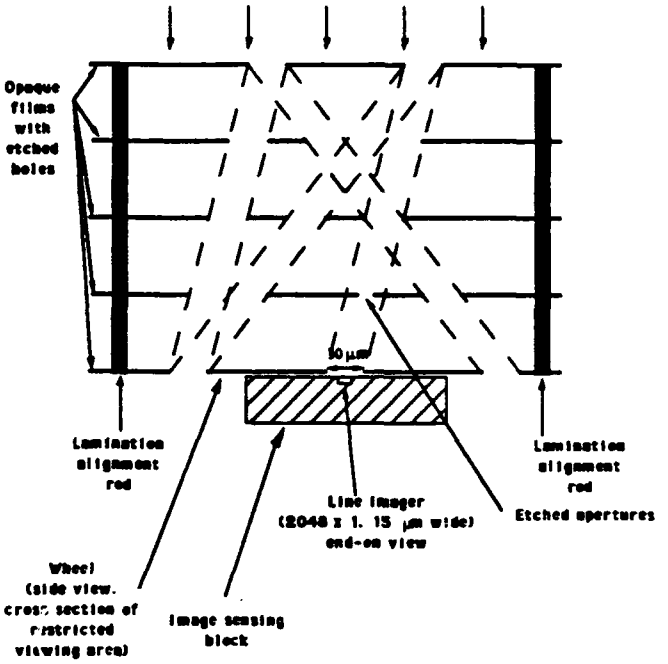


Figure 4. Light paths.

the image row by row as the wheel rotated). Thus, at each instant, the photoconductor was illuminated by the light from only one image pixel. The wheel of home TV was identical to that in the studio. It was synchronized with the studio wheel such that the image would be a reproduction of the scene in the studio.

The Wheel of Fortune sensor design is shown in Figure 3. A rotating laminated glass wheel about 3 cm thick and 16 cm in diameter rotates at 1800 RPH (30 Hz). On one side of this wheel is a camera lens with a low f-number. This lens focuses a scene to be analyzed by the machine vision system onto one face of the wheel as a 3 cm x 3 cm image. This face of the wheel is finished as a fine-grain ground-glass diffusing surface. On the opposite side of the wheel from the lens is a 2048 x 1 line image sensor with 15 micrometer pitch pixels. The sensor is mounted in an image sensing block that rides aerodynamically upon the polished surface of the glass disk at a spacing of less than 1 micrometer.

The glass disk is comprised of 5 or more laminations. Each lamination is an optically flat disk of glass with a thin film of hard opaque material on one side of it. This opaque film is etched using photolithography. The etched pattern consists of a set of circular holes. As shown in Figure 4, these holes are deliberately designed to align with holes on different lamina to form light paths. Each path begins with at a particular location within the image and terminates at a particular sensor pixel. Approximately 1000 paths between the image and sensor simultaneously active at each instant. Each path is approximately 30 micrometers in diameter, except that in the final layer next to the sensor the diameter of path can be reduced to effect a weighting of the amount of light passing to the sensor through the path. By arranging the geometry of the paths so that no spurious paths are produced, many millions of paths can be implemented in a 16 cm diameter disk. Each path corresponds to the operation of multiplicatively weighting one original scene image pixel by a positive value and adding this product to an accumulating sum.

Each pixel in the line array records one component of one image feature that utilizes weights between -1 and +1. For example, one pixel might be configured to receive the weighted pixel inputs needed to calculate the imaginary positive portion of a two-dimensional Gabor logon at a particular position, scale, and orientation [5] [6]. Another pixel might be configured to receive the real negative portion of the same

[5] J. G. Daugman. "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression". *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-36, pp. 1169–1179, July 1988.

[6] J. Buhmann, J. Lange, and C. von der Malsburg. "Distortion Invariant Object Recognition by Matching Hierarchically Labeled Graphs". In *Proc. Intl. Joint Conf. on Neural Networks*, vol. I, pp. 155–159, Washington, D.C., June

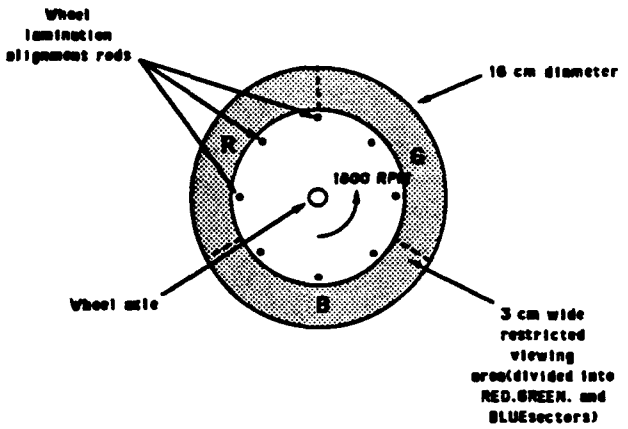


Figure 5. A laminated wheel containing three sectors.

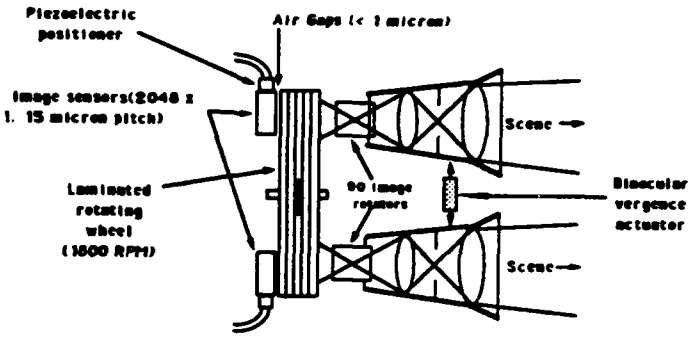


Figure 6. A binocular vision wheel.

logon. If the paths are configured correctly, as the wheel spins each pixel sensor will receive exactly the image data it needs to compute its feature.

Any feature that can be derived from a weighted sum of pixel intensities can be computed. Such features include those required for object acquisition, object recognition, object measurement, local optical flow determination, and object tracking. The glass wheel and image sensor system described above actually computes 3 sets of features per wheel rotation as shown in Figure 5. A third of the wheel's image input face is coated with a red-pass filter just below the ground glass outer surface. The next third is coated with a green-pass filter and the final third is coated with a blue-pass filter. Thus, the features derived by the system are computed separately in each of these three color bands. For other applications, such as for night vision using thermal infrared, infrared-pass filters covering various wavelength bands could also be used, as the silicon sensor could be replaced by one more suited for long-wave infrared operation.

By using two lenses and two line sensors it is possible to achieve binocular vision as shown in Figure 6. The lenses have a vergence actuator/sensor that is used with some specialized features and feedback loop to accurately measure the range to a relatively close object. This passive object range capability can be of great value in applications such as autonomous systems, surveillance systems, and industrial systems for unconstrained environments.

The maximum computational capability of the Wheel of Fortune system is easy to calculate. In the case of the binocular line imager system with 2048 features per line imager, an average of, say 65,536 (256×256) scene image pixels used per feature and 180 images per second, the system will carry out as many as 24.2 Gigaops. This performance is quite impressive given a sensor cost of $100 (the high volume commercial production cost goal).

4 Prototype Demonstration Model

We built a prototype demonstration model to verify the concept of the foveated image sensor. The demonstration model contains 5 laminations. Each lamination is 0.3 cm thick and 19.2 cm in diameter. For demonstration purposes, we used the principle of electromagnetic reciprocity. Instead of installing a sensor array and using a set of images to determine that the correct features were being computed, we installed light sources (light emitting diodes) at the position where sensors would be and verified that the correct features were formed on the "image input" side of the wheel. A Gabor logon wavelet was used as the test feature. The real part of this feature, as shown in Figure 7, was sampled in an 11×11 pixel window and these were used to weight the microscopic holes in the first layer. To eliminate the negative values, weights were shifted and normalized to [0,1]. Figures 8 and 9 shows the microscopic holes in the first and fifth

layers. Note that the microscopic holes in the first layer were weighted with the real Gabor logon wavelet. By placing a ground glass screen in close proximity to the front of the wheel we were able to see the features as the wheel was spun by a battery-powered electric motor at approximately 1800 rpm. These features did indeed have the appearance of the Gabor logon features as shown in Figure 10. This validates the basic concept of the Wheel of Fortune sensor.

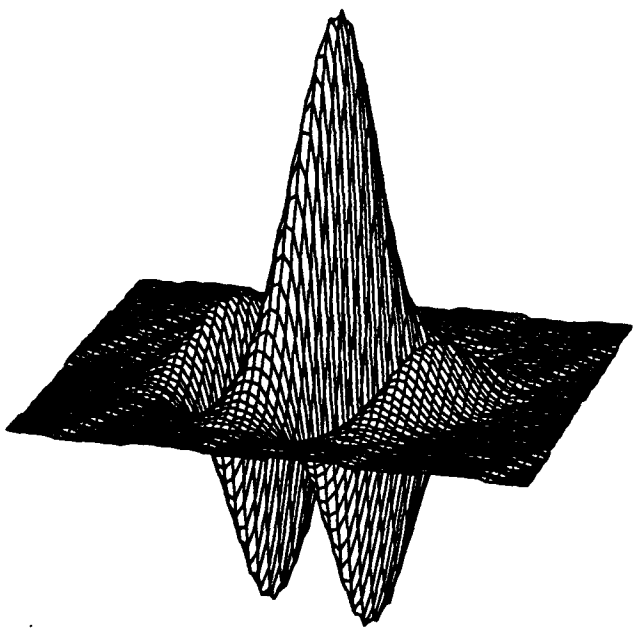


Figure 7. A perspective plot of the real part of a Gabor logon wavelet.

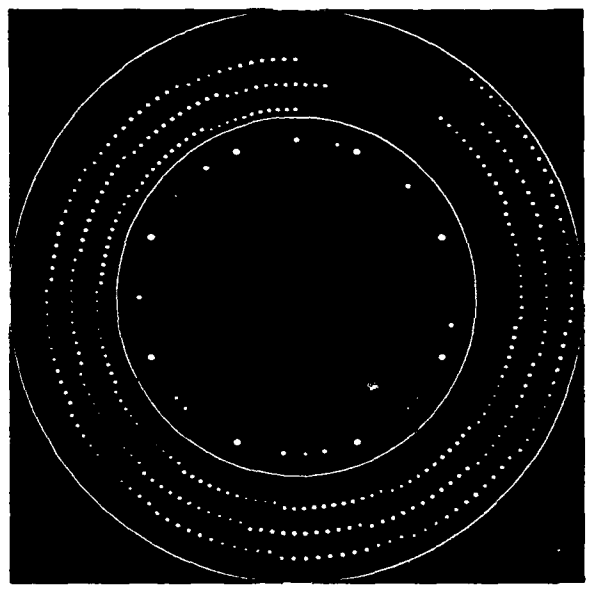


Figure 8. Microscopic holes in the first layer.

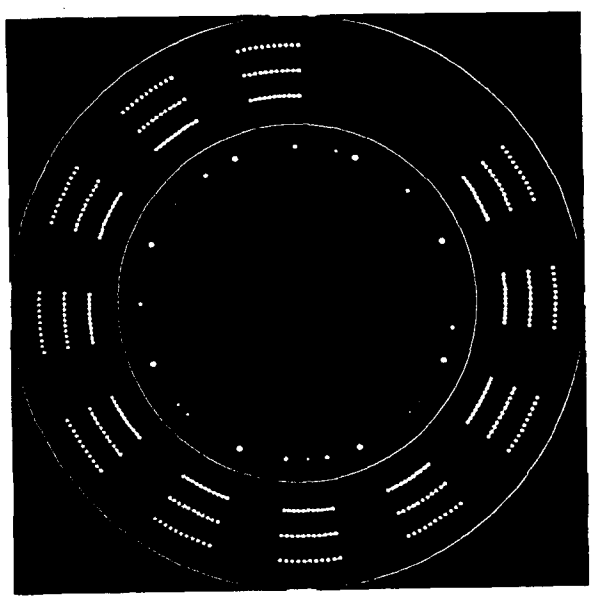


Figure 9. Microscopic holes in the fifth layer.

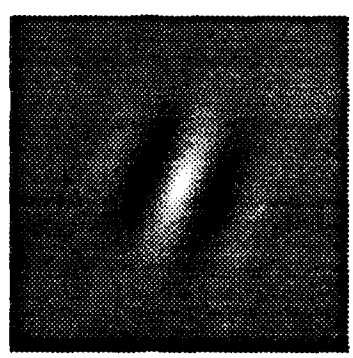


Figure 10. The intensity image of a Gabor logon wavelet, which is the same as that in Figure 7, can be seen in the front of the wheel.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

NEURAL NETWORK DATA FUSION FOR CLASSIFICATION OF EARTH SURFACE FROM SSMI MEASUREMENTS

Y.M. Fleming Lure*, Norman C. Grody+, Y.S. Peter Chiou*, and H.Y. Michael Yeh*

*Caelum Research Corporation, Silver Spring, MD 20901
+National Oceanic and Atmospheric Administration (NOAA), Washington, D.C. 20233

ABSTRACT

Neural network data fusion system is used for fast and accurate classification of five earth surface conditions and surface changes, based on seven SSMI multichannel satellite measurements. The system processes sensory data in three consecutive phases : (1) pre-processing to extract feature vectors and enhance separability among detected classes; (2) preliminary classification of Earth surface patterns at two, separate and parallely acting, classifiers back propagation neural network and binary decision tree classifiers; and (3) data fusion of results from preliminary classifiers to obtain the optimal performance in overall classification. Both binary decision tree classifier and fusion processing centers are implemented by neural network architectures. *The fusion system configuration is a hierarchical neural network* architecture, in which each functional neural net will handle different processing phases in a pipelined fashion.
There are totally around 13,500 samples for this analysis, of which 4% are used as training set and 96 % as testing set. After training, this classification system is able to bring up the detection accuracy to 94 % compared with 88 % for back propagation artificial neural network and 80 % for binary decision tree classifiers. The neural network data fusion classification is currently under progress to be integrated in an image processing system at NOAA and be implemented in a prototype of a massively parallel and dynamically reconfigurable Modular Neural Ring (MNR).

1. INTRODUCTION

Artificial neural networks (ANN), have demonstrated capabilities for robust pattern classification in the presence of noise and object-to-background sensory uncertainty, such as environmental monitoring applications including land cover determination, vegetable mapping, soil survey, etc., or multichannel satellite imagery. This paper presents a neural network data fusion system which will utilize multichannel SSMI satellite imagery, to combine supervised trainable and self-organized neural network architectures with specific knowledge-based classification techniques, with reference to fast and accurate classification of earth surface. This neural approach is intended to compensate different classification techniques by using data fusion method and to reduce the lengthy training time required in supervised learning network. The overall neural network data fusion system, which will be described in more detail, can also be seen as a four-layered supervised network which is composed of several modular and hierarchical networks. In this paper, we will start by a background discussion of the measurement used in this study. The ANN approaches, the BDT approaches implemented by an non-adaptive ANN, and the fusion system will be presented. Hardware implementation of each component in a Modular Parallel Ring (MPR) will also be discussed. Some experimental results will be presented and summary will be given.

2. BACKGROUND

This section briefly review the relevant characteristics and pre-processing for the SSMI data set used in this study. The more detailed description of the SSMI measurements and pre-processing can be found in literatures[1]. The SSMI instrument, flown on board the Defense Meteorological Satellite Program (DMSP) polar orbiting satellites, is a seven-channel conically-scanning microwave radiometer, measuring brightness temperatures at 19, 22, 37, and 85 GHz. All measurements are obtained with dual polarizations (H and V) except for 22 GHz channel. It provides unique signatures for identifying surface features and obtaining the temperature and condition of the Earth's atmosphere. The measurements at different frequencies, different spatial resolutions are convolved to the 55-km resolution of the lowest-frequency channel. This enables each pixel to have seven different measurements without having to consider the effects of spatial inhomogeneity on the different channel measurements. The measurements (brightness temperature or sometimes called antenna temperatures) used in this study were made between November 1988 and January 1989 and covers over the northern hemisphere. The data was identified and confirmed by "ground truth" as five different data sets corresponding to five different surface classes: non-scattering medium (Non-Sm), precipitation over the ocean (R-Ocean), snow cover land (Snow), precipitation over the land (R-Land), and the desert (Desert). Each class has different samples ranging from 500 to 5000 and there are totally over 13,500 samples. Table 1 illustrates some SSMI measurement classification characteristics including SSMI measurements and surface features[2].

3. NEURAL NETWORK DATA FUSION SYSTEM

This system will process sensory data in three consecutive phases, as follows: (1) pre-processing, aimed at extracting feature vectors and in enhancing separability among detected classes; (2) preliminary classification of Earth surface patterns at two, separate and parallely acting, classifiers back-propagation ANN (AP ANN) and binary decision tree (BDT); and (3) fusion of classification results performed at global fusion center (GFC) from different classifiers and imagery to obtain the optimal decision. The configuration is a hierarchical neural network architecture, in which each *functional neural net will handle different processing phases in a piplined fashion.*

3.1 Pre-processing

Pre-processing for SSMI imagery includes mainly the generation of (7 x 7) covariance matrices from measured brightness temperatures at each pixel. Information about pixel brightness temperatures, covariance matrix elements, and desired surface class definitions, is collected in a feature vector for the supervised training of a neural network classifier. It has been demonstrated that increasing the elements of the feature vector by adding more relevant parameters, derived nonlinearly from original features, can reduce the number and size of hidden layers, and can also reduce the training time [Marks, et al, 1988]. Since the covariance matrix evaluation involves the manipulation of two matrices, the operations involved are suitable to neural network implementation by feed-forward topologies, by merely assigning two manipulated matrices to the weights and input vectors of the back propagation neural architecture, as has been investigated.

3.2 Preliminary Classification

a. BP ANN Classifier

A two-layered (one hidden layer) supervised back propagation algorithm is used to train the network to become a feed forward pattern recognition engine[3] to learn the input feature vectors corresponding to different output classes.

---

[1]Grody, N.C., 1991, "Classification of Snow Cover and Precipitation Using the Special Sensor Microwave Imager, J. Geophy. Res., Vol. 96, No. D5, 7423-7435.

[2]Lure, Y.M.F., N.C. Grody, Y.S.P. Chiou, and H.Y.M. Yeh: "Classification of Earth Surface from Special Sensor Microwave Imager (SSMI) Using Artificial Neural Network (ANN) Data Fusion," IGARSS '92, May 26-29, Houston, TX.

[3]Rumelhart, D.E., J.L. McClelland, "Parallel Distributing Processing: Explorations in the Microstructure of Cognition," MIT Press, Cambridge, MA, 1991.

There are 14 input neurons corresponding to SSMI measurements as well as their covariance matrix, 60 hidden neurons, and 5 output neurons representing 5 surface conditions. It takes around 40 and 160 iterations of presenting training set to train the BP ANN classifier up to 90% and 100%, respectively.

b. BDT Classifier

The BDT classifier is constructed to implement Grody's global classification algorithms as in Figure 2.[1] They are designed to analyze global coverage of satellite data sets and to classify based on the physical characteristics of measurements and on surface types. This technique performs a hierarchical tree-structured decision procedure through the evaluation of polynomial functions of input feature elements and through thresholding. The special topology of BDT classifiers, used for surface condition classification based on SSMI measurements is drawn from the so-called Entropy Net architecture[4]. This architecture includes a two-layered topology, of which the lower layer performs arbitrary mapping of thresholding operations while the upper layer performs logical operations (e.g. AND, OR) which allow us to convert the hierarchical decision procedure into a fully parallel process. The weight vectors between the layers are determined from the coefficients of polynomial functions. A striking advantage of the proposed neural implementation architecture is that it allows us to specific the number of neurons needed in each layer, along with the desired output. This, in turn, leads to an accelerated progressive training procedure, that also allows each layer to be trained separately.

There are 5 neurons corresponding to 4 SSMI measurements as well as one element of covariance matrix and 5 output neurons for each surface class. The individual decision from both BP ANN and BDT modules are sent to the global fusion center (GFC) for final decision.

3.3 Fusion Processing

The fusion processing involves global fusion center (GFC) operations, which integrates results from both BP ANN and BDT classifiers. The GFC is composed of several different data fusion center (DFC), each of which corresponds to different types of output classes. A self-adjusted or self-trained learning algorithm is used in each DFC to set the optimal decision rules such that the total probability of detection is maximized. This data fusion scheme, also called distributed-detection scheme, corresponds to a two-layered network of nonlinear threshold elements (e.g., binary or sigmoidal functions). Their weights of these elements are trained by using off-line stochastical information to form detection system. The stochastic information including priori probability, probability of false alarm, and of missing detection is obtained by comparing classification results from individual classifiers with ground-truth data. The approximation rules are obtained from the nonlinear combination of the statistics of previous classification results from individual classifiers[5]. The DFC can be further implemented by using a self-adaptive neural net such that its weights and bias are calculated based on the analogue output values of BP ANN and BDT for each SSMI measurement.

4. IMPLEMENTATION PROCEDURE

The procedure for implementation of neural network data fusion can be summarized as follows:

1. Preparation of training set.
2. Separation of training set into several groups such that the clustering center of each class among same group are easy to be separated.
3. Selection of neural classifiers such as BP ANN and BDT preliminary classification.
4. Training of preliminary classification processor to a satisfactory accuracy.
5. Evaluation of the results from preliminary classification to obtain the detection statistics.
6. Determination of weights and bias in each fusion center from the detection statistics.

[4]Sethi, I.K.: "Entropy Nets: From Decision Trees to Neural Networks." Proc. of the IEEE, Vol. 78, No. 10, October 1990.

[5]Tenney, R. R., and N. R. Sandell, "Detection with distributed sensors", IEEE Trans. Aeros. Elec. Sys., pp 501-509, 1981.

5. CLASSIFICATION RESULTS

There are totally 13,787 samples of data used in this study. Each of five different classes contains from 500 to 5,000 different samples. We used 500 samples of data as training sets which represent 3.6% of the total samples. Each training set, obtained randomly from the total data set, consists of equal amount of samples from five different classes. Rest of the samples (over 96%) are used for testing the network. Once the BP ANN is trained either fully or partially, it is used to perform the classification. The classification accuracies, by using the fully-trained BP ANN classifier (i.e., all training patterns are recognized by this BP ANN), are 82%, 98%, 97%, 78%, and 79% for non-scattering medium, precipitation over ocean, snow, desert, and precipitation over land, respectively (Lure, et al., 1992). The classification accuracies are 99%, 56%, 81%, 57%, and 70% for each surface class. Note that the class of non-scattering medium represents the surface which can not be easily, specifically identified as any other four surfaces. The overall accuracy for BP ANN approach is around 88% whereas it is around 80% for BDT classifier. The preliminary results show that the neural network data fusion system improve the classification accuracy for all classes by around 4 % from BP ANN's results. The overall accuracy of neural network data fusion is improved to 94 %. Even without fully-trained (e.g., 90% of training set are learned correctly by BP ANN) the overall classification accuracy can still maintain similar classification accuracies. From the coefficients of data fusion center, it is also found that BP ANN plays more important role in classifying the non-scattering medium, snow, and desert; whereas the BDT are more dominant in classifying the other two surfaces. The significance of each SSMI measurement to classification of each of five surface types can also be obtained through the linearlization procedure of the weights described in the previous study.

6. SUMMARY

In this research effort, neural network data fusion system is presented to classify surface types based on the SSMI measurements. Both back propagation ANN (BP ANN) and binary decision tree (BDT) classifier are used for this study. Seven SSMI measurements (brightness temperature at 19, 22, 37, and 85 GHz for H and V polarizations, except H for 37 GHz) at each image pixel are extracted as an input feature vector. Five surface types including non-scattering medium, precipitation over the ocean, snow cover land, precipitation over the land, and the desert are used as target patterns. After trained by using less than 4% of the samples, both BP ANN and BDT are able to perform the classification over 13,500 samples. The overall accuracy for BP ANN and BDT approaches are 88% and 80%, respectively. The neural network data fusion system which fused the individual decision from BP ANN and BDT improved the overall accuracy to 94%. The significance of contribution from either approach is determined based on the coefficients of data fusion center. The fusion system is currently implemented in a massively parallel and dynamically reconfigurable neural network hardware (Modular Neural Ring) for real time parallel processing[6] and integrated in an image processing system at NOAA/NESDIS. The hybrid classification system not only preserves the advantages of both BP ANN and BDT classifiers (for example, the capability of physical interpretation of input feature space from the BDT classifier and robust classification from the BP ANN) and but reduce the pitfall of individual classifier (for example, brute-force training of the BP ANN module and sensitive to noise for the BDT module).

ACKNOWLEDGEMENTS

This work is partially supported by NOAA/NESDIS and Air Force Rome Development Center under SBIR Phase II contract: F30602-89-C-0130. The SSMI data set were provided by Hughes Aircraft Co. through NOAA/NESDIS. The authors are grateful for the discussion with P. A. Ligomenides and L. Jump of Computer Systems and Architecture Group at Caelum Research Corporation.

[6]Ligomenides, P. A., L. B. Jump, and Y-S. Chiou, 1991: "A Reconfigurable Ring Architecture for Large Scale Neural Networks", Proc. of Conf. Fuzzy and Neural Syst. and Vehi. Appli. 1991, Tokyo, Japan.

Table 1 SSMI classification characteristics

Channel frequencies and polarizations							
SSMI Measurements	TH(19)	TV(19)	TV(22)	TH(37)	TV(37)	TH(85)	TV(85)
Surface Conditions	Non-SM		R-Ocean	Snow		R-Land	Desert

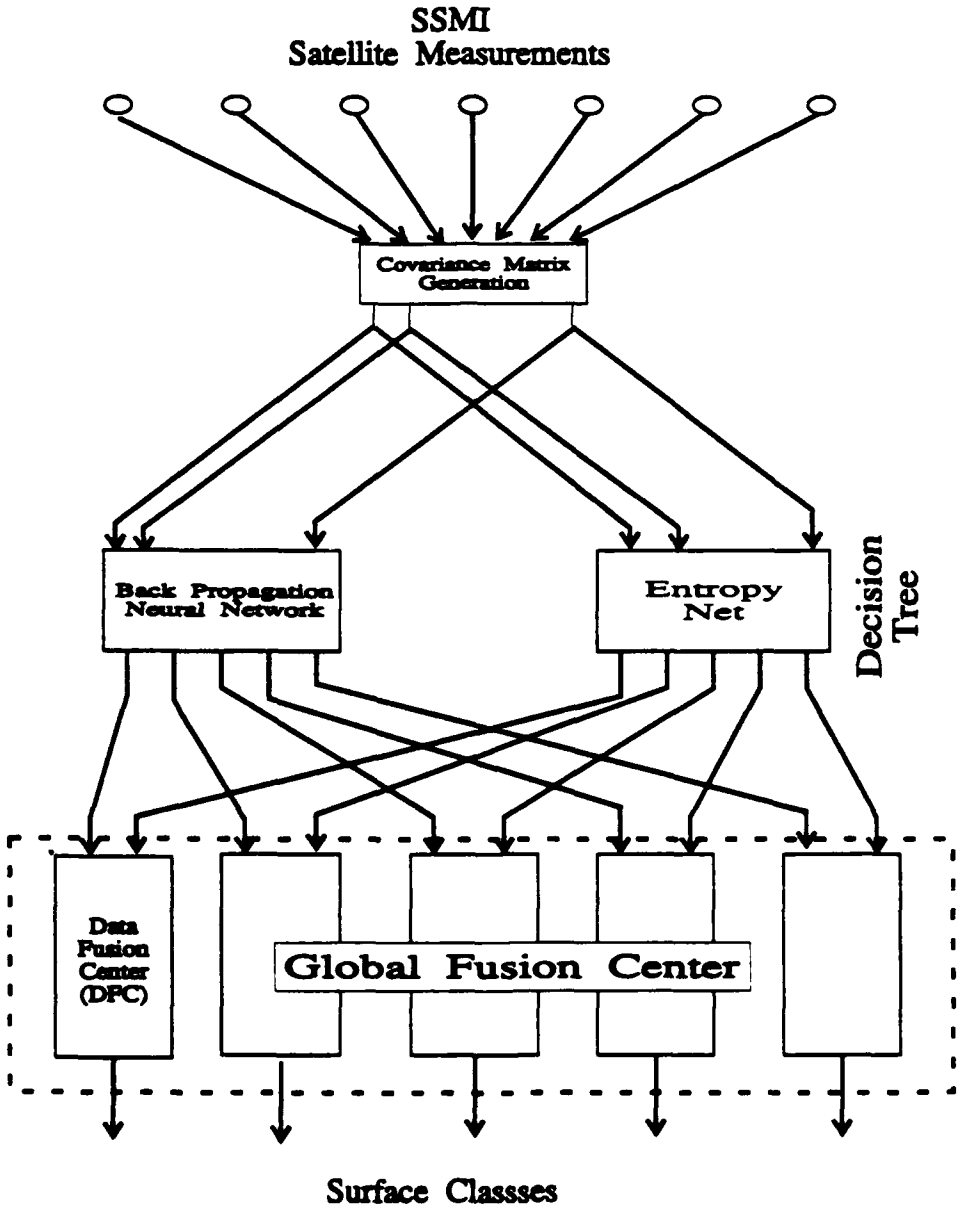


Figure 1. Neural Network Data Fusion System for SSMI Measurements

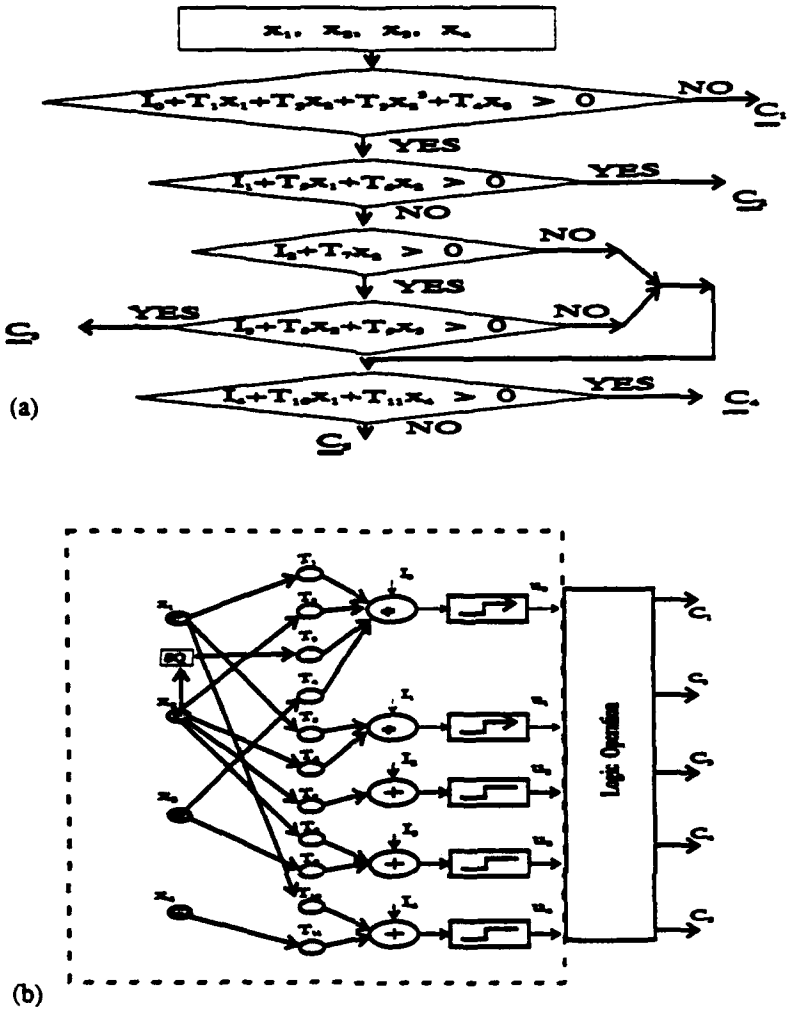


Figure 2. BDT Classifier and its Neural Implementation

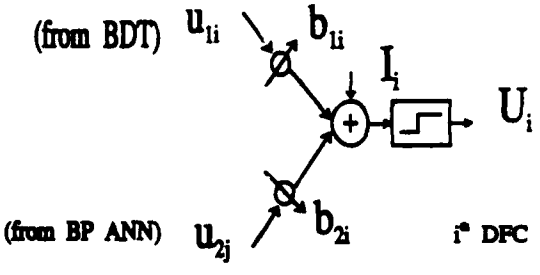


Figure 3. Data Fusion Center

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

CLASSIFICATION OF MULTISPECTRAL DATA: A COMPARISON BETWEEN NEURAL NETWORK AND CLASSICAL TECHNIQUES

July 1992

J.H. Seldin and J.N. Cederquist
Optical and Infrared Science Laboratory
Advanced Concepts Division
Environmental Research Institute of Michigan
P.O. Box 134001, Ann Arbor, MI 48113-4001

Abstract

Two neural networks were used for single-pixel classification of an agricultural scene imaged by a multispectral scanning sensor. Data collected in five wavebands from the near-infrared to the short-wave infrared were used to train and test multi-layer perceptron and Kohonen neural networks. The ability of these networks to discriminate between five terrain classes found in an agricultural scene was shown to be equal to or better than a classical maximum-likelihood classifier, with the Kohonen network demonstrating the best classification performance of the three techniques.

1.0 Introduction

Multispectral sensors add a third dimension to conventional imaging by supplementing 2-D spatial information with a spectral response in a number of selected wavebands. This type of data allows discrimination of classes via their spectral signature without requiring a description of their spatial properties. Single-pixel discrimination relies solely on a characterization of spectral signatures, and a judicious selection of wavelengths with which to image can exploit the differences in spectral signatures between classes of interest. The development of multispectral classification algorithms that do not rely on spatial characteristics is of interest, and we discuss here the results of implementing two distinct neural networks to classify a specific set of terrain types found in an agricultural scene.

Section 2 briefly discusses the two neural networks, the multi-layer perceptron and the Kohonen self-organizing, that were applied to this problem. Section 3 describes the sensor and the spectral wavebands used, and the agricultural scene of interest and the classes therein are shown. The results of classifying the scene with the neural networks and a statistically-based maximum-likelihood classifier are presented in Section 4, and the classification experiments are summarized in Section 5.

2.0 Neural Network Overview

Two neural networks, the multi-layer perceptron and the Kohonen self-organizing feature map, were applied to this problem. The multi-layer perceptron (MLP) is a well-known fully-connected network consisting of processing nodes that pass the weighted sum of the inputs to that node plus a threshold through a nonlinear function, in this case a sigmoidal nonlinearity. The back propagation training algorithm is used with MLP networks that have continuously differentiable node nonlinearities. It is a least-mean-squares gradient search algorithm that

minimizes the mean-squared difference between the actual and desired outputs of the network by propagating the error at the output back through the network, adjusting node weights and thresholds appropriately. The back propagation algorithm is supervised; it performs a search based on the difference between the actual output and the desired output. There are versions of this iterative training algorithm for which the training is accelerated, but no such variant was used in this work. A very basic description of the MLP and the back propagation algorithm is found in Ref. 1.

Very different from the MLP is the Kohonen self-organizing feature map, which has been applied to pattern classification and vector quantization [1]. As the name implies, the Kohonen network learns in an unsupervised mode, clustering inputs with common features. Upon convergence, node weights tend to specify clusters in the output that are distributed according to the probability density function of the input training vectors [2]. The network consists of a single output layer that is fully connected to the input layer, and every output node has an associated neighborhood of other output nodes, each at a given distance. For a given input, the network finds the winning output node, defined as that node with weights that are closest to the input with respect to some metric, typically Euclidean distance. During training, the weights of the winning node and its topological neighbors are updated and another input is presented. This proceeds in an iterative manner with the neighborhoods slowly shrinking in size until each neighborhood has only a single output node; i.e., only the weights of the winning node are updated. After training, the weight vectors for an output node tend to resemble the mean of the class represented by that node. The training procedure for the Kohonen network is discussed in more detail in Refs. 1 and 2.

The non-parametric neural network classifiers were compared to a model-based maximum-likelihood (ML) classifier. The ML classifier makes a Gaussian-distributed assumption and requires estimates of the mean vector and covariance matrix to compute the class-conditional density functions. For our case, the means and covariances for each class are formed through supervised training. Once trained, the ML classifier assigns to a pixel the class with the underlying Gaussian density which maximizes the probability of the pixel occurring given that class.

3.0 Description of the Multispectral Data

The neural networks discussed in Section 2 were applied to the problem of classifying terrain types from image data collected by the ERIM M7 scanning multispectral sensor. The M7 scanner is an airborne passive sensor that collects data in 16 wavebands spanning the ultraviolet, visible, near-infrared, and the short-, mid-, and long-wave infrared. Along with signal returns from the ground, the sensor collects calibration data for each waveband. The calibration information enables the removal of sensor bias as well as the conversion of M7 signal counts to scene radiance. To make the multispectral classification experiments more realistic, we restricted ourselves to using only a subset of the available bands. A deployed sensor typically would not have the luxury of, or in many cases would not require, the large number of bands available from the M7 sensor. We chose for agricultural classification the 5 bands in Table 1. These bands were chosen to exploit the differences in spectral responses of vegetation and soil types. A different set of bands could have been selected for other classification problems.

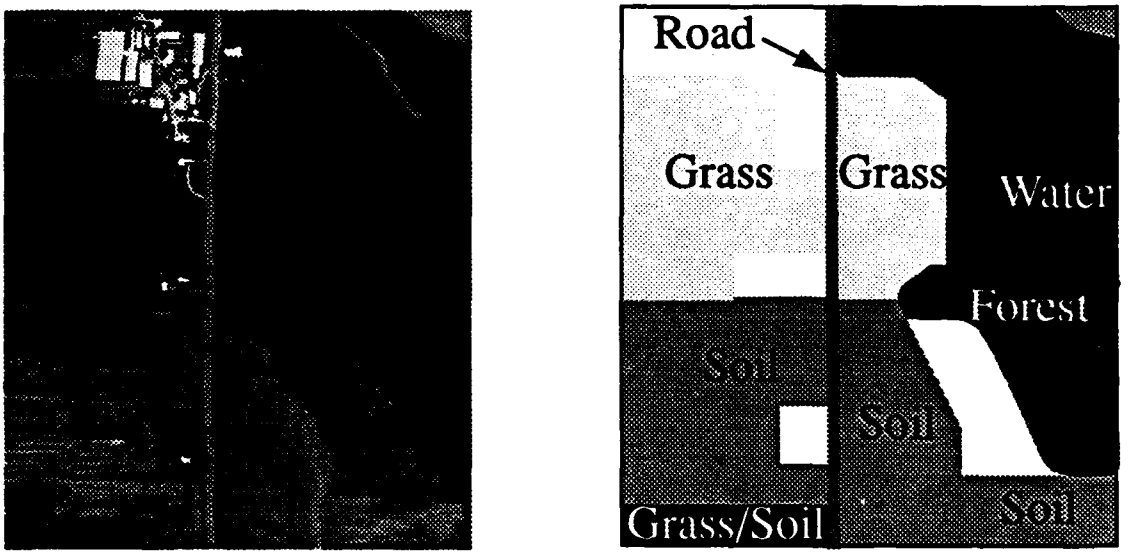
Channel Number	50% Response (On-Off μm)	Bandwidth (μm)
8	0.646 - 0.723	0.077
9	0.811 - 0.938	0.127
11	1.230 - 1.296	0.066
12	1.560 - 1.665	0.106
13	2.012 - 2.290	0.278

Table 1: M7 wavebands used for multispectral classification.

A standard procedure was adopted for preprocessing the M7 data. The first step is to subtract the sensor

bias measured during the collection for each channel. After bias subtraction, the data for each channel are scaled such that 95% of the data in that channel fall in the range [0,1] after scaling. This scaling accommodates the neural networks, which are typically trained with data scaled to roughly this range. The scaling information was obtained from a post-collection histogram of each channel, but could be done in real-time if necessary by adjusting the sensor gain to match the sensor gain used during the collection from which the training data were obtained.

An agricultural scene over Coldwater, MI was selected for the multispectral classification experiments. A 700 x 600 square-pixel portion of this scene in the 0.494-0.531 μm (green visible) band is shown in Fig. 1 along with ground truth. The sensor followed the road down the middle of the scene, collecting returns from both man-made and naturally-occurring areas. The ground truth for this scene is in areas quite good; in general the fields observed in this region consisted primarily of vegetation (weeds, alfalfa, grass) or they were mostly plowed, exposing the soil. The ground truth in Fig. 1(B) shows the primary classes in the image and their approximate location. The blank regions in this map are areas for which there is no ground truth available, or which contain man-made objects, such as buildings, for which there is limited data compared to the larger natural regions.



(A) (B)

Figure 1. M7 agricultural scene over Coldwater, MI. (A) Image in visible (0.5 μm) band; (B) Ground truth.

4.0 Multispectral Classification Results

The primary classes found in the Coldwater scene are shown in Fig. 1(B): grass, soil, forest, water, and road. The other major class in the scene is buildings, but these vary greatly depending on the type of material used for the roof, the angle of the building relative to the sensor, etc. We chose to train the multispectral classifiers to discriminate between these 5 classes for which there is an ample amount of both training and testing data.

The selection of training regions followed several self-imposed guidelines. We first decided to use only a small fraction of the available data for training the classifiers. The training set consisted of 1000 pixels from each class; the resulting 5000-member training set represents only 1% of the data in the scene. The training data was taken from rectangular regions, where pixels from the same class are likely to be similar to one another. In this case the inter-class variance of the examples possibly could be smaller than the variance of all the pixels of that class located throughout the entire scene. Another characteristic of the training set is that the representative rectangular regions from each class were chosen as close to the horizontal center of the image as possible. The sensor roughly followed the road down the center of the image, so pixels near the center of the image are closest to the sensor, and the pixels at the edge of the image where the aspect angle is large are farthest from the sensor. Pixels

at the edge of the image, therefore, are more likely to suffer from atmospheric path-length attenuation, and these pixels could be more difficult to classify correctly. By choosing the training regions near the center of the image where path-length is at a minimum, we are not giving the classifiers any advantage when classifying pixels at the edge of the image.

The Kohonen self-organizing network topology is defined by the problem: we have a 5-class problem using data from 5 spectral bands. The Kohonen network consists of a single layer connecting the 5 input nodes, corresponding to the number of bands, to the 5-node output layer, corresponding to the number of classes. A linear output-node topology was selected for the 5 x 5 network; that is, output nodes are connected in a line, where every node but the two end nodes has two neighbors. Because of this, the output node weights that are most similar after training tend to be nearest neighbors. The MLP input and output topology is similarly defined by the problem, but the number of hidden layers and nodes in these layers must be determined. It has been shown that no more than three layers are required to from arbitrarily complex decision regions, but in practice two layers are usually sufficient. We experimented with a number of two-layer networks, and we were satisfied with the performance of the simplest of these networks: a 5 x 5 x 5 two-layer perceptron.

The three classifiers were trained with the 5000-pixel training set, and were then applied to the entire Coldwater scene. The resulting classification maps are shown with the ground truth in Fig. 2. A threshold was not applied by any of the classifiers due to the difficulty in selecting a single threshold for each classifier that yielded the same false alarm rate in every classifier for all the classes. Also, although the ground truth for this scene is good on a large scale, it is difficult to establish the pixel-by-pixel ground truth, and so the false alarm rate would be difficult to accurately measure. The 5 x 5 Kohonen network classification in Fig. 2(B) is the best of the three classifiers. The grass regions are classified nearly perfectly, as are the areas of soil. The lake and the river feeding it are clearly recognized, and the road is correctly classified, but with a smaller width than observed in the image. In some areas the shoulder of the road was not assigned to the road class. The forest area is not homogeneous, mixing areas of soil and scattered grass pixels. It is difficult to determine the exact ground truth for this region, so it is not clear whether these areas of soil are incorrect. Several small clumps of forest are classified along the road, presumably from single trees. Scattered throughout the classification map are pixels classified as road where one would not expect to find this class. These misclassifications tend to be in areas near soil, and an examination of the mean spectral signatures estimated for these two classes reveals a high degree of similarity. These two classes provide the most difficult discrimination test for all the classifiers. Overall, however, the Kohonen classification is very accurate.

The MLP classification in Fig. 2(C) is accurate in the water areas and in the grass region to the right of the road. Classification of the forest region is more uniform than for the Kohonen network, and the road width (including the shoulders) is more consistent with the data than for the Kohonen classifier. However, the MLP has difficulty distinguishing soil from road. We observe that soil pixels in the lower left region of the scene are often mistaken for road. This is a region in which there is quite a bit of variability in the soil that was not captured in the training set. The same type of inhomogeneity is found in the grass region to the left of the road, and we see similar misclassifications. The worst classification of all is from the ML classifier [Fig. 2(D)]. The regions where it makes mistakes are similar to those for the MLP; however, it makes mistakes far more often. The estimates of the means and covariances for each class did not sufficiently capture the variability in areas outside of the training regions. Also, a histogram of the road and water classes reveals that a Gaussian assumption may not be appropriate for these classes. The ML classifier performs well in the forest area, but it cannot handle most of the soil areas and does poorly in much of the grass regions.

5.0 Summary of Classification Results

The neural network classifiers met with excellent success for the 5-band multispectral classification of the Coldwater agricultural scene. Of the three classifiers, the Kohonen network demonstrated the best classification performance. The Kohonen network achieved near-perfect classification of the grass, soil, road, and water, and very good performance in the inhomogeneous forest region. The MLP and ML classifiers also performed well, but frequently misclassified portions of the soil areas as road. Variability in both the soil and grass regions that was not captured in the training set caused problems for these two classifiers, but the Kohonen network was more

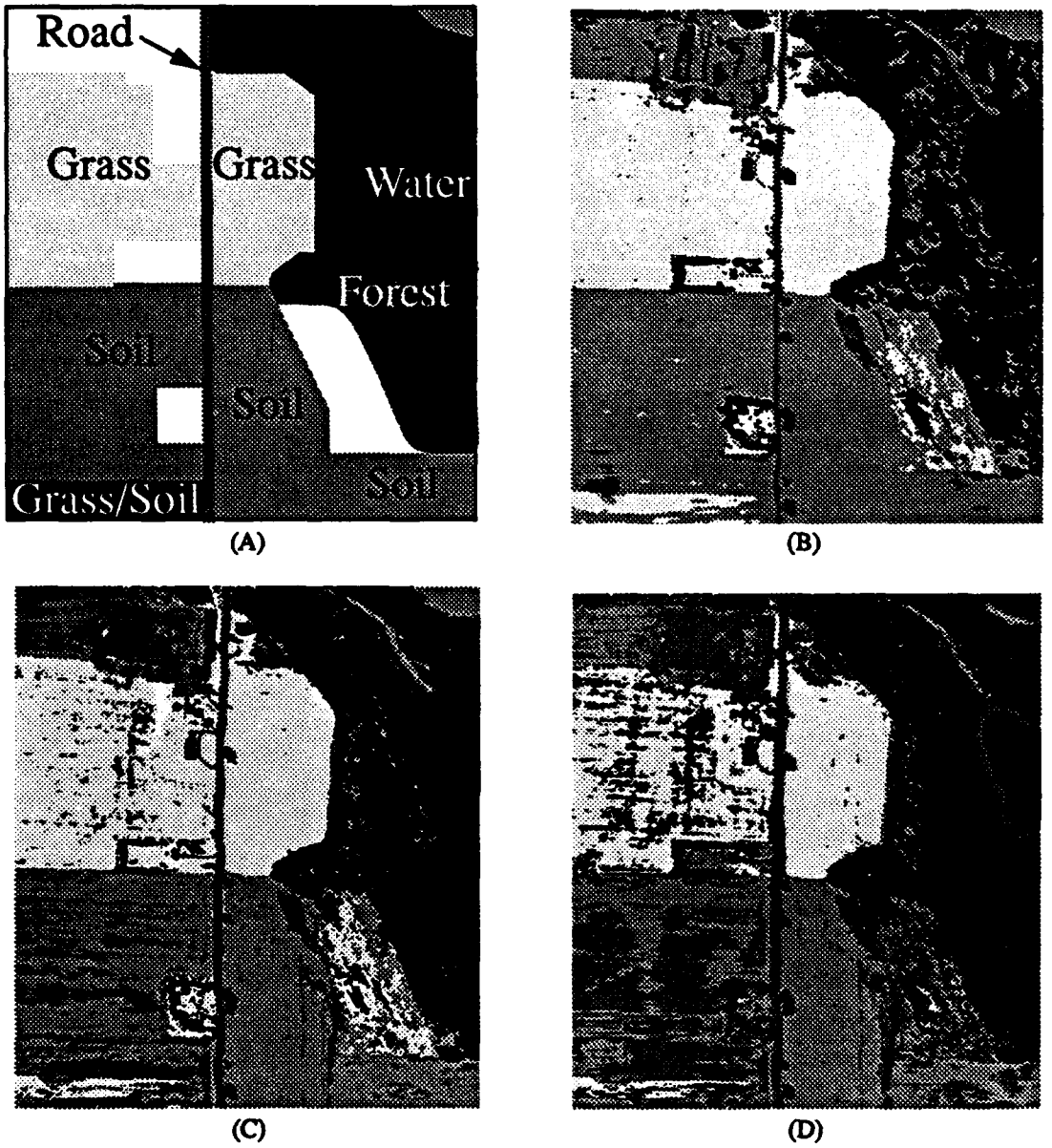


Figure 2. Multispectral classification results. (A) Ground truth; (B) 5 x 5 Kohonen network; (C) 5 x 5 x 5 multi-layer perceptron; (D) Maximum likelihood.

robust with respect to these variations. The Kohonen network trains more quickly than the MLP, is unsupervised, and requires fewer computations during classification that do the other two classifiers. Overall, the performance of the MLP was slightly better than that of the ML classifier. The ML classifier might be improved with more training data, or by splitting terrain types with large variability into more than one class. Large amount of training data for estimating means and covariances is not always available, however, which makes the neural networks more attractive.

6.0 References

[1] R.P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, p. 4, April 1987.

[2] T. Kohonen, "Self-Organization and Associative Memory," Springer-Verlag, Berlin, 1984.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

TRACK BEFORE DETECT USING MODEL BASED NEURAL NETWORK (U)

24 -26 August 1992

Leonid I. Perlovsky
Nichols Research Corporation
Wakefield, MA 01880

(U) ABSTRACT

(U) A new approach to multiple target tracking in heavy clutter is developed. This approach utilizes Bayesian classification techniques for associating data on multiple scans. Efficient implementation of this approach is made possible by using the Maximum Likelihood Adaptive Neural System's (MLANS) unique feature of combining neural networks and model based techniques. The approach is equally applicable to active sensors, IRST, or multiple sensors.

(U) We have developed a novel approach to tracking multiple objects in heavy clutter based on a previously developed MLANS neural network[1,2,3] . In our approach, the MLANS estimates target and track models in real time and performs a fuzzy classification of all targets in multiple frames into multiple classes of tracks and random clutter. This novel approach to tracking results in a dramatic improvement of performance: the MLANS tracking exceeds performance of existing tracking algorithms, it is capable of track initiation by considering multiple frames, it eliminates combinational search via fuzzy associations, and it is utilizing track models as well as target feature models for data association and track initiation and maintenance.

(U) Historically, intrinsic mathematical similarities between tracking and classification problems have not been explored. Tracking problems have been characterized by an overwhelming amount of data available from radar sensors. This has led to the development of suboptimal algorithms amenable to sequential implementation for handling high rate data streams. Such algorithms based on Kalman filters converge to the optimal, Maximum Likelihood (ML) solutions, however, their generalization to multiple object tracking has been difficult[4]. Classification and pattern recognition problems, on the other hand, have been characterized by insufficient amount of data for unambiguous decisions, which has led to the development of Bayes classification algorithms, optimally utilizing all the available information.

(U) Such optimal utilization of information is lacking in existing tracking algorithms. This leads to difficulties with tracking multiple objects in heavy clutter. As a number of clutter returns increases, it is becoming increasingly difficult to

---

[1]Perlovsky, L.I., "Multiple Sensor Fusion and Neural Networks," *DARPA Neural Network Study*, MIT Lincoln Laboratory, Lexington, MA, 1987.
[2]Perlovsky, L.I., "Neural Networks for Sensor Fusion and Adaptive Classification," *First Annual International Neural Network Society Meeting*, Boston, MA, 1987.
[3]Perlovsky, L.I. and McManus, M.M., "Maximum Likelihood Artificial Neural System (MLANS) for Adaptive Classification and Sensor Fusion, *Neural Networks*, 4(1) pp. 89-102, 1991.
[4]Blackman,S.S., Multiple Target Tracking with Radar Applications, Artech House, Norwood, MA, 1986.

(U)
maintain and especially to initiate tracks. A near optimal algorithm, the Multiple Hypothesis Tracking (MHT)[5], initiates tracks by considering all possible associations between multiple objects and clutter returns on multiple scans. This problem, however, is known to be NP complete[6], its optimal solution requires combinatorially large amount of computation, which is difficult to handle even for neural networks, when a number of clutter objects is large. A partial solution to this problem is offered by the Joint Probability Density Association (JPDA) tracking algorithm[7], which performs fuzzy associations of objects and tracks, eliminating combinatorial search. However, the JPDA algorithm performs associations only on the last frame using established tracks and is, therefore, unsuitable for track initiation.

(U) The MLANS neural network explores mathematical similarity between tracking and classification problems and applies optimal Bayes methods to the problem of tracking multiple objects. By applying fuzzy classification to associating data in multiple scans and the ML estimation of track parameters the MLANS combines the advantages of both the MHT and the JPDA. In MLANS tracking, each track is a class characterized by the state parameters and by the track model. For example, linear tracks are characterized by their initial positions, velocities, and a linear model relating target coordinates over time to their current positions. Target models are characterized by features such as amplitude or shape of returned pulses. The parameters of these models are estimated adaptively, in real time. In addition to the track state parameters, each class-track is characterized by its covariance matrix, which can be adaptively estimated or fixed using prior knowledge about tracks and sensor accuracy. Utilization of track and target models within the neural network architecture leads to fast learning approaching the information-theoretic limit established by Cramer-Rao bounds. The MLANS performs the optimal estimation of track parameters, utilizing all the available returns from several scans. This optimal utilization of all the available information makes the MLANS ideal for track initiation, as well as for track maintenance in heavy clutter. The MLANS is applicable to both, coherent, radar tracking as well as to incoherent imaging sensor data.

(U) In the example below, the MLANS is applied to tracking three objects in random clutter, using simulated pulse-Doppler search mode radar data. The scattered plot in Figure 1 shows the distribution of returns from 10 scans in the Doppler-velocity vs. range coordinates. About 1000 random clutter returns appear distributed throughout the plot, while three clusters of returns corresponding to the moving objects have approximately 100 returns each. Due to a wide beam employed in a search mode, there is approximately 10 returns from each object per scan. The scatter seen in object returns along the Doppler velocity is determined by the radar accuracy, while the scatter along the range axis is due to both, the radar accuracy and the motion of the objects. Each return is characterized by its time and by its 4 coordinates: range, Doppler velocity, elevation and azimuth angles. The MLANS is using time as a parameter of the track models to project each track to its current position. By clustering these data in the 4-dimensional coordinate space, the MLANS initiates and estimates track state parameters which are shown in Figure 1 by illustrating 10-σ ellipses of the estimated parameters. The accuracy of this estimation is determined by the Cramer-Rao bound due to the ML estimation performed by the MLANS.

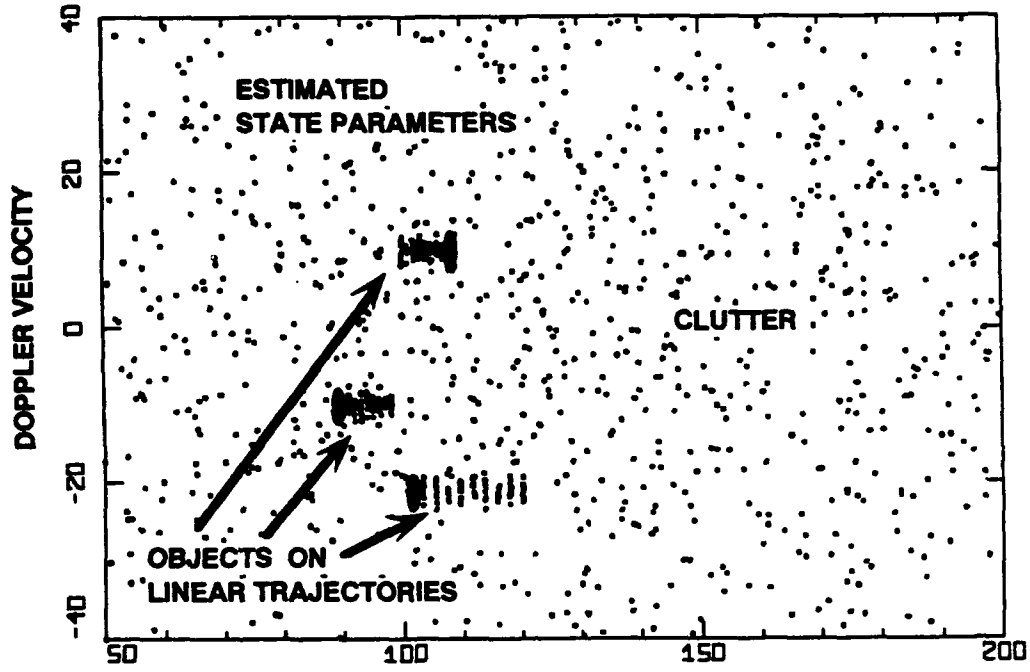
(U) Let us examine now *the MLANS association of tracks and radar returns*. In this example, the MLANS maximum number of allowed tracks was 5. The MLANS performs fuzzy association by calculating a probability P(k|N) for each return N to belong to track k. These probabilities are shown in Figure 2. It is seen that three tracks (1, 2, 3) have nearly 100% probabilities for all returns from the three objects with very little clutter contribution, two tracks (4 and 5) acquire just a few clutter returns and are terminated, and nearly all the clutter returns are identified as such with 100% probability. The three tracks acquiring real objects are declared detections and their trajectory parameters are output by the MLANS. Each of these tracks acquire ~1 clutter return, or 0.1% of 1000 total clutter returns: this number corresponds to the best possible performance; it is the true overlap (the Bayes error) between distributions of clutter and track returns for this case. The MLANS achieved the bounds of the Bayes errors for accuracy of association and Cramer-Rao bounds for accuracy of track parameters.

---

[5]Singer, R.A., Sea, R.G., and Housewright, R.B., "Derivation and Evaluation of Improved Tracking Filters for Use in Dense Multitarget Environments," *IEEE Transactions on Information Theory*, **IT-20**, pp. 423-432, 1974.

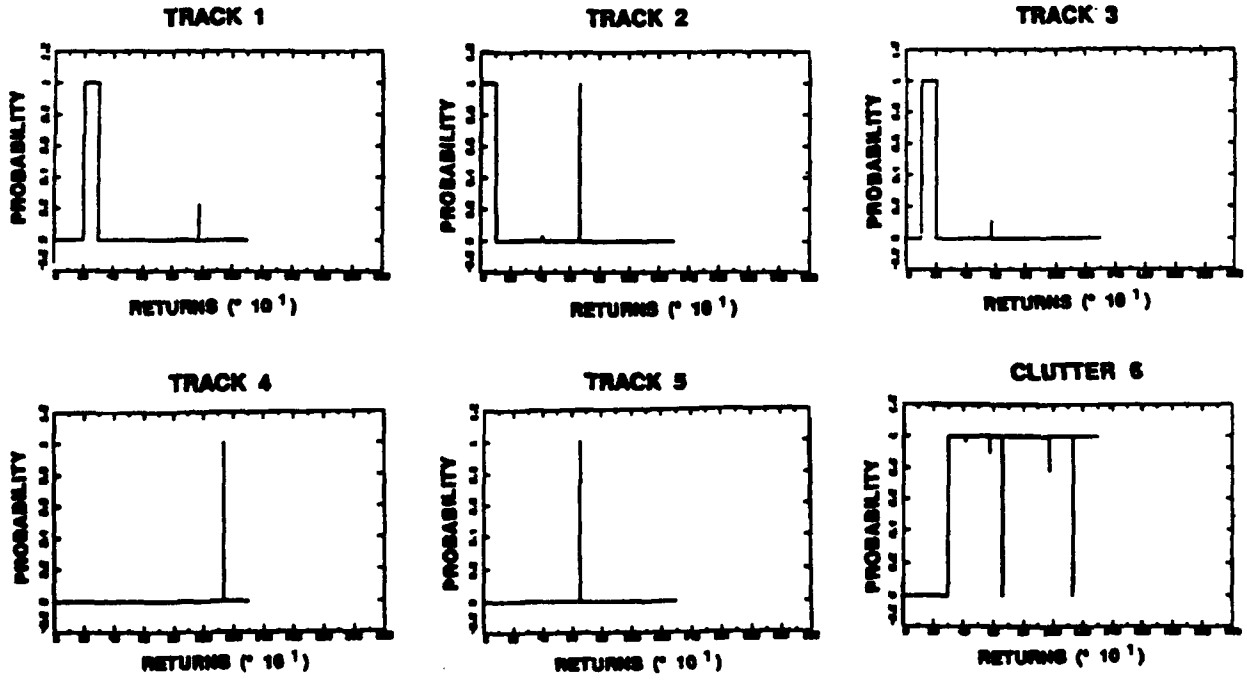
[6]Parra-Loera, R., Thompson,W.E., and Akbar,S.A., "Multilevel Distributed Fusion of Multisensor Data," Conference on Signal Processing, Sensor Fusion and Target Recognition, SPIE Proceedings Vol. 1699, 1992.

[7]Bar-Shalom, Y. and Tse, E., "Tracking in a Cluttered Environment with Probabilistic Data Association," *Automatica*, 11, pp. 451-460, 1975.



(UNCLASSIFIED)

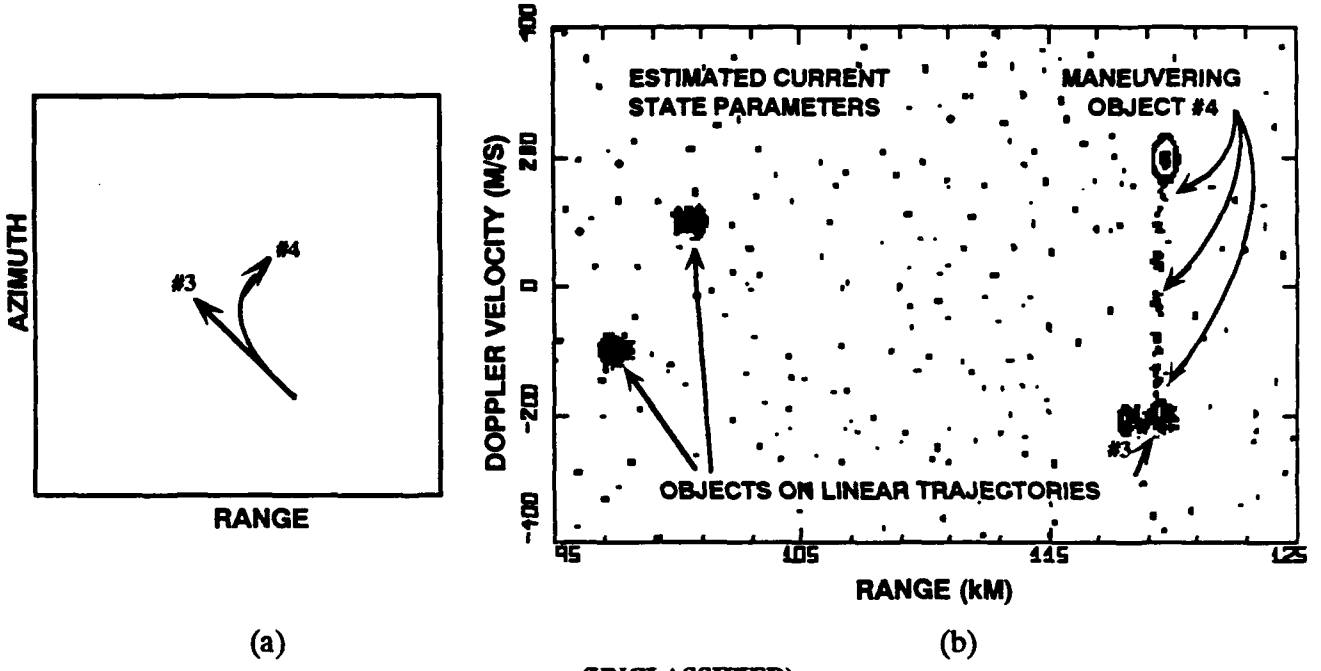
Figure 1. (U) The MLANS Combined Detection and Tracking.



(UNCLASSIFIED)

Figure 2. (U) The MLANS Probabilistic Associations

(U) In the next example, Figure 3, the MLANS is applied to tracking four objects in random clutter. The objects #3 and #4 initially occupy same range and Doppler cells, making initiation of tracks difficult for existing algorithms. Our results indicate that the MLANS successfully initiate tracks for crossing and maneuvering objects.



(UNCLASSIFIED)

Figure 3. (U) The MLANS Tracking Maneuvering and Crossing Objects

(U) Data correlation and tracking capabilities of the MLANS described above can be extended to multiple sensors in a straightforward way: the fact that data come from a single sensor is not essential in the above described example. The problem of sensor fusion is a difficult one for the application of standard optimal Bayesian inference because the statistical distributions of the data are not known exactly. The absence of knowledge of the prior probabilities has been long recognized as a difficulty for the Bayesian approach. In reality, however, the problem is not limited to the absence of knowledge of the priors, but in fact all the distribution parameters are usually unknown, and the common assumption of Gaussian distributions is not usually valid. The MLANS capability for adaptive classification that was essential for tracking is also useful for sensor fusion via adaptive estimation of object distributions[3]. The MLANS is also useful in complicated cases when the information to be fused originates from sensors of different types, having different coverage or field-of-views (FOV), or operated asynchronously. Data sets resulting from such observations are often incomplete: while many objects are observed by a single sensor, only a few may be observed by all the sensors. The MLANS fuses such incomplete data sets, using a recently developed technique described in Perlovsky and Marzetta[8].

---

[3]Perlovsky, L.I. and McManus, M.M., "Maximum Likelihood Artificial Neural System (MLANS) for Adaptive Classification and Sensor Fusion, *Neural Networks*, 4(1) pp. 89-102, 1991.

[8]Perlovsky, L.I. and Marzetta, T.L., "Estimating Covariance Matrix from Independent Incomplete Realizations of a Random Vector," accepted for publication in IEEE Trans. on Signal Processig, 1992.

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

AN ARCHITECTED NEURAL NETWORK FOR CONTACT STATE ESTIMATION: A COMPARISON TO THE CRAMER-RAO LOWER BOUND

Christopher M. DeAngelis, Kelly J. Ross
Naval Undersea Warfare Center
Division Newport, Code 222
Newport, Rhode Island, 02841 USA
e-mail: deangelis@ada.nusc.navy.mil
conant@ada.nusc.navy.mil

Robert W. Green
University of Massachusetts Dartmouth
Computer and Information Science Department
North Dartmouth, Massachusetts, 02747 USA
e-mail: green@cis.umassd.edu

ACKNOWLEDGMENTS

The authors would like to take this opportunity to thank their manager William Navin who, since 1988, has provided continuing encouragement, support, and advice on exploring neural networks and their application to combat control systems. This project was supported by the Office of Naval Technology; sponsor Dale Howser ONT-232.

Abstract - A long-standing method for contact state estimation involves physically marking a surface with bearing lines and then systematically looking for contact tracks that satisfy predetermined "speed strip" constraints; namely, given a contact moving with constant velocity and heading, it is constrained to traverse equidistant points over equal time intervals. The Neurally Inspired Contact Estimation (NICE) system has been developed to automate this method. NICE is a fixed-architecture artificial neural network inspired by the biological retina system. The marking surface is represented by an input layer of receptor neurons; presentation of bearing lines over this field provides stimulation to the underlying receptors. Subsequent neural layers are selectively interconnected to impose the speed strip constraints of constant velocity and heading. The output neural layer derives the bearing, range, course, and speed values associated with the most likely contact track. This paper presents the NICE method, discusses potential usage, and compares the NICE method with the Cramer-Rao lower bound.

1. INTRODUCTION

1. Contact State Estimation Problem Domain

The general contact state estimation, or target motion analysis, problem is to estimate contact location and motion from available sensor readings. In a general sense, each sensor reading provides constraints on the contact state. For example, a line-of-sight bearing reading of 305 degrees at time 0700 constrains the contact to be somewhere on a line northwest from the observer's location at 0700. If sufficient observations are available, including at least one observation platform maneuver, and if assumptions are made about the contact motion (such as constant speed and heading), then the contact state may be constrained to only one possible solution[1,2]. In this case, the contact state is said to be

---

[1] S.C. Nardone & V.J. Aidala, "Observability Criteria For Bearings-Only Target Motion Analysis", IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-17, No. 2, March 1981.

[2] K.F. Gong, S.E. Hammel, S.C. Nardone & A.G. Lindgren, "Three dimensional Contact Parameter Estimation", Proceedings of the Sixteenth Asiloma Conference on Circuits, Systems and Computers, November 1982.

observable. A great deal of work has been done to determine the circumstances under which various aspects of contact state are observable. The important point here is that contact state estimation is a constraint problem.

Due to the uncertainty, or error, associated with physical sensor readings, contact state determination is indeed a parameter estimation problem[3,4,5]. Even though the contact scenario may be fully observable, noisy sensor readings will preclude an exact solution to contact state (i.e. the true solution). A method must be employed to determine the most likely estimate of contact state. Typically, mathematical estimation techniques such as least squares or maximum likelihood are employed, and are based on some measure of compliance between actual and hypothesis-predicted observations. A related issue is solution sensitivity: Given that the observations are noisy, which non optimal contact state solutions are above a certain degree of likelihood? Furthermore, are the almost-optimal solutions tightly or loosely clustered in the contact state space?

2. Speed Strips Method

The speed strips method has been a long-standing technique for estimating contact state. This method is illustrated in figure 1, which shows a two-leg observer scenario with line-of-sight bearings to a contact observed at equal time intervals.

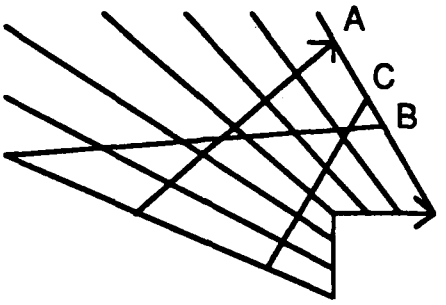


Figure 1. Speed Strip Fitting

If one assumes that the contact maintains constant heading and speed during this period, one can search for the contact by laying a straight-edge across the bearing lines and moving it around until a placement is found wherein the distance traversed between observations is proportional to the time between observations. In this example, path A satisfies the "equal distance between equal observations" constraint. Note that the magnitude of the distance on path A (i.e. the length of the strip) is proportional to the speed of the contact, hence the method name of speed strips. For comparison, note that paths B and C violate the constraint, and are not valid contact paths.

Speed strips obviously implement the constraint nature of contact state estimation. With noisy observations, the operator would manipulate the speed strips to find a most likely estimate of contact path since no path would likely fit exactly. Solution sensitivity would be evident in how far the speed strip could vary from the best solution while still fitting reasonably well to the bearing lines.

2. NICE: AN ANN FOR CONTACT STATE ESTIMATION

Our biological neural networks (human brains) are able to process diverse information ranging from sensory inputs to higher order reasoning. The scenario in figure 1 was presented to the reader in a visual manner. Can one find the best solution by visual processing alone? Not easily, or speed strips wouldn't be necessary. The input layer of neurons in the human vision system feed into hidden layers that extract various correlations in the raw image, edges of figures for

---

[3] J.G. Baylog, A.A. Magliaro, S.M. Zile & K.F. Gong, "Underwater Tracking in the Presence of Modeling Uncertainty", Proceedings of the Twenty-First Asilomar Conference on Signals, Systems and Computers, November 1987.

[4] K.F. Gong, J.G. Baylog & A.A. Magliaro, A Decision-Directed Approach to Solution Integration for Tracking in an Underwater Environment", Proceedings of the Eighteenth Asilomar Conference on Circuits, Systems and Computers, November 1984.

[5] C.M. DeAngelis & R.W. Green, "Constructing Neural Networks for Contact Tracking", Neural Networks for Signal Processing - Proceedings of the 1992 IEEE Workshop.

example. Note that an edge is just a line segment in the visual field where on one side receptor neurons are more excited by more light, and on the other, less excited by less light. In contrast, humans do not appear to have hidden layer neurons that correlate visual input and detect paths that are evenly spaced between bearing lines!

The fact that humans do not possess speed strip constraint processing neural networks does not preclude construction of such artificial neural networks (ANNs)[5]. In considering such a network, the first question is: How should the inputs be represented? A simple way is to represent the two dimensional ocean surface of interest as a square grid (or plane) of input neurons with a separate plane allocated for each bearing observation time. A given input neuron will have an activation of '1' if a bearing line traverses the ocean grid cell it represents (i.e. stimulates it), and '0' otherwise.

The next question is: How could the correlation layer be configured to embody the speed strip constraints? A straightforward way is to have a correlating neuron for each possible contact path. Each path would be defined by the ocean grid cells in which it begins and ends. The correlation neuron would be connected to the appropriate ocean grid cell for each of the bearing observation times; that is, the neuron's inputs are the ocean cells the contact would be in if it were on the path being represented. The simplest neuron function for these correlating neurons is a boolean AND over all its inputs. A correlating neuron's activation will be '1' if all its inputs (which do not need to be weighted) are '1', and '0' otherwise. Essentially, a correlating neuron will be active if all bearing lines were crossed at the appropriate locations and times for the track it represents.

Due to the finite input grid resolution, such a network would activate not only the correlation neuron corresponding to the true contact path, but nearby contact paths as well. Thus, a set of correlation neurons would be active in the network and another layer of neurons might be needed to determine the best estimate for the contact state attributes of bearing, range, course, and speed. Figure 2 shows the architecture of the ANN described in this section.

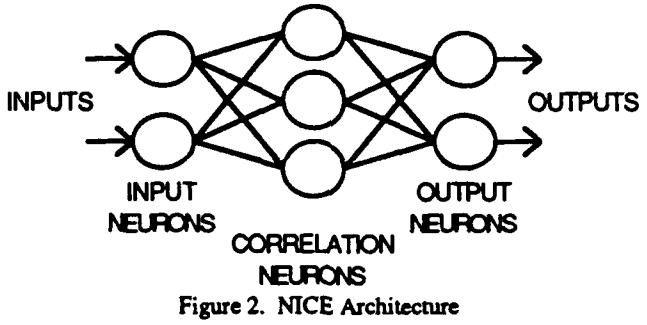


Figure 2. NICE Architecture

Note that the model described thus far is very simple: rectangular input grids, boolean marking of input cells, and boolean activation of correlating neurons. Clearly, more sophisticated alternatives exist. Polar-coordinate input planes may better reflect the nature of line-of-sight bearings and bearing errors. Input layer cells might be marked by their probability in relation to the bearing measurement error. Correlation cells could be real-valued and approximate a least-squares or maximum-likelihood function. There are certainly many possible enhancements to the simple Neurally Inspired Contact Estimation (NICE) model.

3. SYSTEM PERFORMANCE

1. A NICE Scenario

Figure 3 presents an observer-contact scenario, where the observer traverses two legs. At eight times, noise-free passive measurements are made. Assuming propagation mode, contact depth, and ocean bottom depth, the contact must be at some position on a hyperbola (for this particular sensor) at successive measurement times. With the contact maintaining a constant course and speed, the speed strip model holds, and a NICE ANN can be used to determine contact path.

An evolutionary processing of bearings by NICE is shown in figure 4. Each path displayed corresponds to a correlation neuron that is active. After four observations, the contact state is not observable since the contact could be either right or left of the observer. Successive observations after an observer maneuver resolve the left-right ambiguity and lead to an increasingly well defined solution set.

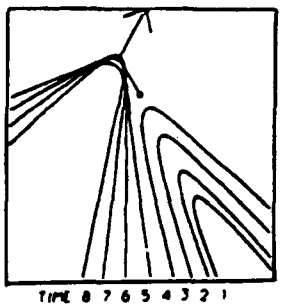


Figure 3. Observer-Contact Scenario

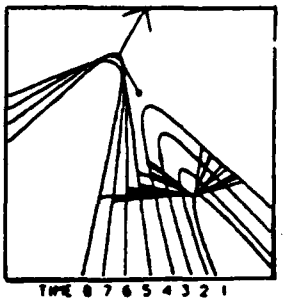
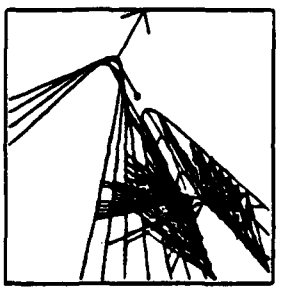


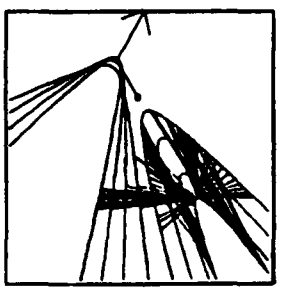
Figure 4. Evolutionary Processing of Bearings

2. Grid Resolution and The Effects Of Noise

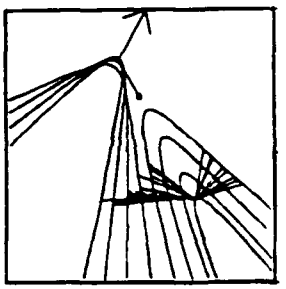
For noise-free bearings, increased grid resolution means increased solution resolution, as seen in figure 5. In this figure, the same scenario is run through NICE at resolutions of 64, 128, and 256 cells per edge. As the resolution increases, the sharpness of the solution set increases. Note, however, that the averages for range, course, and speed are nearly the same for all three resolutions. It would appear that the highest possible grid resolution should be used, but it must be remembered that the processing required in the correlation neurons goes roughly as the square of the grid resolution. Thus, there is a trade-off between solution sharpness and processing time.



(a) 64



(b) 128



(c) 256

Figure 5. Grid Resolution

For noisy observations, the situation is more complex. Remember that, as in the manual speed strip discussion, noisy measurements will preclude exact fits. If a high resolution grid is used with noisy data, no correlation neurons will fire because their linked input neurons are likely to include at least one cell that is not marked since measurement error caused the observed bearing line to miss the cell. However, as grid resolution is decreased, each grid cell covers a larger bearing range, and thus the appropriate cells are likely to be marked, even by a noisy observation. Indeed, as bearing noise increases, grid resolution must be decreased to find contact paths. Note that, though solution quality decreases with noise level, NICE continues to produce good results.

3. Comparison With Cramer-Rao Lower Bound

The Cramer-Rao lower bound is a common benchmark used in parameter estimation which places a lower bound on the variance of an unbiased estimator. Two observer-contact scenarios were presented to NICE and system performance was compared to the above benchmark[6]. These comparisons were performed by examining monte-carlo simulations for the given geometries at various noise levels, The first geometry was poorly observable, and the second moderately. Normalized mean error and standard deviation values for the estimates of range and bearing are summarized in table 1 and table 2.

---

[6] D.J. Ferkinhoff, C.M DeAngelis, K.F. Gong, S.E. Hammel & R.W. Green, "On Training Artificial Neural Networks for Tracking in the Ocean Environment", Proceedings of Oceans '92, Newport, RI, October 1992

Noise Level	0.5 σ_o	1.0 σ_o	1.5 σ_o	2.0 σ_o
NICE Range	0.070	0.060	0.014	0.006
NICE Bearing	0.130	0.150	0.200	0.210

(a) Mean Error

Noise Level	0.5 σ_o	1.0 σ_o	1.5 σ_o	2.0 σ_o
NICE Range	0.123	0.229	0.335	0.420
CRLB Range	0.029	0.059	0.089	0.119
NICE Bearing	0.510	0.680	0.920	1.090
CRLB Bearing	0.147	0.294	0.441	0.587

(b) Standard Deviation

Table 1. Geometry 1: Normalized Statistics

Noise Level	0.5 σ_o	1.0 σ_o	1.5 σ_o	2.0 σ_o
NICE Range	-0.089	-0.027	-0.088	-0.103
NICE Bearing	0.190	0.110	0.280	0.240

(a) Mean Error

Noise Level	0.5 σ_o	1.0 σ_o	1.5 σ_o	2.0 σ_o
NICE Range	0.052	0.100	0.128	0.175
CRLB Range	0.030	0.061	0.091	0.122
NICE Bearing	0.430	0.490	0.750	0.900
CRLB Bearing	0.135	0.269	0.404	0.538

(b) Standard Deviation

Table 2. Geometry 2: Normalized Statistics

As can be seen, NICE provides comparable performance, especially at higher noise levels. The means are approximately zero, and the standard deviations approach the CRLB.

4. SUMMARY AND CONCLUSIONS

NICE offers a number of attractive features. Since a bearing line constrains the locus of points where a contact might be at a given time, different angle-of-arrival sensors merely produce different loci; all are equivalent and can be fused using NICE. Intermittent data can be accommodated by configuring correlation neurons to ignore the missing data, and the receptor resolution can be varied to adjust to the quality of the sensor readings, fine for accurate and coarse for noisy. In addition, the neural network can be executed in a highly parallel manner. Overall, NICE has the potential to function as a viable contact state estimator.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

CLASSIFICATION OF LONG SIGNAL SEGMENTS

6 July 1992

Bert de Vries, Ronald Sverdlove, Scott Markel, John Pearson and S.Y. Kung*

Computational Science Group and
the National Information Display Laboratory at
David Sarnoff Research Center
Princeton, NJ 08540
corr.: jpearson@sarnoff.com

*Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

ABSTRACT

In this paper we discuss some neural network design issues for a classification problem involving signal segments of long duration (approximately 100 to 1,000 samples). Some of the drawbacks of traditional neural net memory structures for long signal segments are mentioned and we introduce the gamma memory model, which is particularly suited to store information over long delays. Our problem involves classification over a very large class cardinality (> 1,000) and we discuss the impact of this on the neural network architecture and training scheme. In this paper no experimental results are discussed.

1•0 INTRODUCTION

We are looking into the following problem. Our task is to detect and determine the source (or class) of special signal segments from a relatively low-noise background signal. We will label such a special signal segment as a *signal-of-interest* or abbreviated as *Sol*. The problem is mainly characterized by two difficulties. In the first place, the duration of the discriminating features of *Sol*'s can be quite long, typically between 100 and 1,000 time samples, and may vary largely between classes and even within a class of *Sol*'s. Secondly, the number of different classes for this problem is very large, usually greater than 1,000 categories, and to complicate matters, new signal classes may be added *after* the classification system has been installed.

Detailed domain knowledge about the discriminating characteristics of the *Sol*'s is not available. However, we do have access to a large database of representative examples. From these considerations we decided to implement the essential part of our system in an adaptive neural network.

In this paper, we describe our system design where particular attention is paid to deal with the aforementioned difficulties. At this time, no experimental data and results are released.

2•0 COMPUTING OVER LONG DELAYS IN NEURAL NETWORKS

Consider the following scenario. Let $u(t)$, t=1,...,T, where T is rather large, be a signal-of-interest. $u(t)$ is a member of one of L classes. We intend to design a computational network structure that takes $u(t)$ as input and produces L signals $y_i(t)$, i=1,...,L, as outputs. The activity $y_i(t)$ holds a current (at time t) probability measure that $u(t)$ is a member of class i. Ideally, if $u(t)$ belongs to class i, then $y_i(T)$ is high whereas $y_j(T)$, $j \neq i$, have low values. It is clear that $y_i(T)$, i=1,...,L, is a (non-linear) function of the entire history of $u(t)$, that is, $y_i(T) = f_i(u(t; 1 \leq t \leq T); w)$, where w holds a set of parameters.

In order to achieve this computational functionality, nodes in the network ought to be able to compute functions over the history of activations of (other) nodes. Now let us abstract the classification problem to two communicating nodes in a neural net (see Figure 1(a)). We want $y(t)$ to compute a parametrized function of a history trace of $u(t)$. A popular network structure which implements such a function makes use of a *feedforward tapped delay line*, that is,

$$y(t) = \sum_{k=0}^{K} w_k u(t-k) . \qquad eq.1$$

The tapped delay line structure is depicted in Figure 1(b). For simplicity we have omitted the possible non-linear relation between $u(t)$ and $y(t)$. In this formulation, the history function is parametrized by the weights w_k, k=0,...,K. The number of weights grows proportionally with the memory depth. For long signal segments, that is, for large T, we need a deep memory which translates into a large K, the number of taps or *memory order*, and therefore the number of weights will be large as well. From the statistical literature we know that overparametrizing a function leads to increased error variance, which causes decreased performance on a generalization data set (Geman et al., 1992).

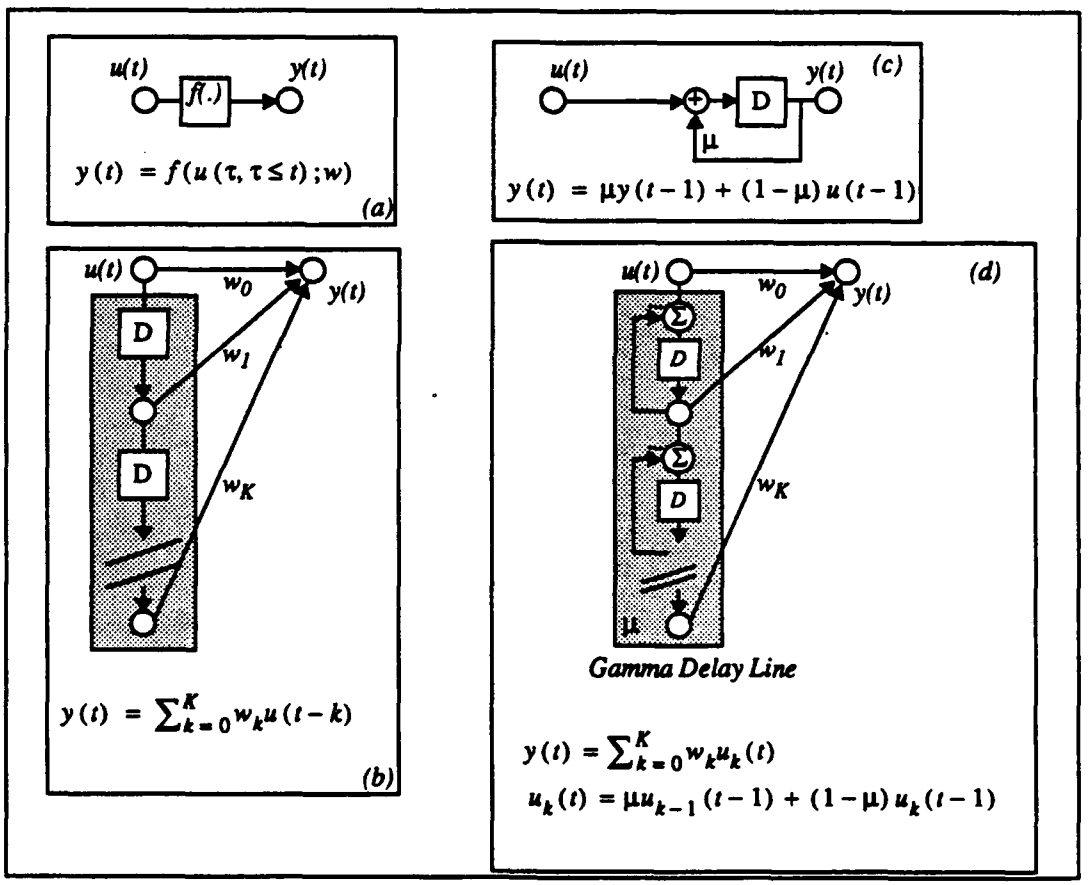


Figure 1• (a) y(t) is a function of a history trace of input u(t). (b) A common network structure makes use of the feedforward tapped delay line. For this structure the memory depth D is fixed and equals the memory order K. The resolution R is fixed and equals 1 time step. (c) The leaky integrator computes a recency gradient over u(t). Here D=1/(1-μ) and R=1-μ. (d) The gamma memory, a tapped delay line of self-recurrent nodes combines favorable properties of both the feedforward tapped delay line and the self-recurrent node. For the tapped gamma delay line, we have a parametrized memory depth D=K/(1−μ) and resolution R=1−μ.

An important question looms: Can we design a computational network structure with variable memory depth while keeping the number of weights constant? In other words, can we *uncouple* memory depth from memory order? The answer, of course, is yes (otherwise we wouldn't have brought it up). Variable memory depth can be produced by recurrent networks. An important example concerns the *first-order self-recurrent node* or *leaky integrator*, which is implemented by (see Figure 1(c))

$$y(t) = \mu y(t-1) + (1-\mu)\,u(t-1)\,. \qquad eq.2$$

Let us compute the mean memory depth of self-recurrent nodes. Observe that eq.2 can be written as

$$\begin{aligned} y(t) &= (1-\mu)\sum_{k=0}^{t-1}\mu^{t-1-k}u(k) \\ &= \sum_{k=0}^{t-1} g(t-k)\,u(k) \end{aligned} \qquad eq.3$$

where $g(t) = (1-\mu)\,\mu^{t-1}$. In this framework, $y(t)$ can be interpreted as computing a history trace of $u(t)$ where $g(t)$ is the (normalized) *memory kernel*. We compute the *mean memory depth* of the memory kernel $g(t)$ as

$$D = \sum_{t=0}^{\infty} t\,g(t) = (1-\mu)\sum_t t\mu^{t-1} = \frac{1}{1-\mu} \qquad eq.4$$

Thus, the memory depth of the leaky integrator increases for increasing values of μ (<1). The case μ=0 reduces the leaky integrator to a unit delay operator, whereas μ=1-ε with ε very small leads to a very deep memory. The cost of increasing memory depth is a reduced temporal resolution. If we define the resolution R as the number of taps (or state variables) per time step delay, then the resolution for the leaky integrator is R=1/D=1-μ.

Let us recapitulate this section. In our problem we want to compute parametrized decision functions on signal segments $u(t)$, t=1,...,T, where T is large. Feedforward tapped delay lines bring along an excessive number of free parameters (proportional to T), which leads to suboptimal performance as the decision function will be over-parametrized. The leaky integrator can modify its memory depth but it has only one variable $y(t)$ available to store a history trace. In order to be able to independently adapt memory depth and resolution, we need at least two adaptive parameters. The leaky integrator however has only one adaptive parameter (μ) available to adapt the two memory functions depth and resolution.

	depth	resolution
tapped delay line (order K)	K	1
leaky integrator (decay μ)	$1/(1-\mu)$	$1-\mu$
gamma memory (order K, decay μ)	$K/(1-\mu)$	$1-\mu$

Table 1• Relations between memory parameters order K, decay μ and memory characterizations depth and resolution.

An adaptive memory structure such as the *gamma tapped delay line* is best suited for our problem. The gamma delay line consists of a cascade of leaky integrators with the same decay parameter μ (Figure 1(d)). The gamma delay line generalizes the tapped delay line and the leaky integrator into one parametrized structure. As a result, the gamma delay line has two parameters available, *memory order* K and *decay* μ, in order to control depth and resolution. Thus, an uncoupling between depth and resolution can be achieved. Table 1 displays how

the memory functions depth and resolution are related to the memory parameters for the various memory network structures. More detailed information on designing neural networks for temporal processing and in particular on the gamma neural network can be obtained from de Vries (1991) and de Vries and Principe (1991).

3•0 TRAINING STRATEGY

The second issue of this paper concerns a training strategy that handles a large class cardinality (> 1,000). Our first design choice is that we create a new network for each signal-of-interest. This in contrast to one neural network that detects and classifies a large number of *SoI*'s. Advantages of the former architecture include incremental trainability and capacity. If we have a net which looks for more than one *SoI* then each time we adapt for a particular *SoI*, the prototypes for all other *SoI*'s are distorted. Obviously, there is a limit to how many different *SoI* prototypes one net can contain before the "cross-talk" destroys everything. So if we want to add another *SoI* class to the vocabulary, *all SoI* prototypes are altered, and performance may drop drastically for *SoI*'s *that were previously learned*. This problem can be avoided if we use one net per *SoI*. When we add a *SoI*, we add a net, and we can design a training strategy that does not alter *SoI* prototypes when that is not desired.

Assume that we have 500 classes and 50 exemplar sequences per class available for training. For any class, this implies that we have only 0.2% positive examples and 99.8% negative examples. If we would use the common *approximation-based* training strategy where the desired signal is high (say one) for a positive example and low (zero) for every negative example, than the desired signal for each of the output nodes equals zero in 99.8% of the training runs. Very good performance on the training data set (99.8% accuracy) would be obtained if we force all outputs to zero at all times, by for instance setting all weights to zero and the thresholds high.

Obviously, this training algorithm doesn't work for large class cardinality. Therefore we use a *decision-based training* strategy, which effectively reduces the cardinality of the training set for each class. Decision-based training, much akin to the original perceptron learning paradigm, proceeds as follows:

1. *if the correct network wins, do not adapt any weights. (In other words, do not change a winning team).*
2. *if net i wins, but net j is the correct net (desired to win) then adapt as follows:*
   a. *adjust the weights for network j so as to increase its output next time for this input pattern, and,*
   b. *adjust the weights for network i so as to decrease its output next time for this input pattern.*

This strategy has been tested extensively at Princeton University in S.Y. Kung's laboratory on a variety of classification problems and indicates better performance than approximation-based adaptation procedures.

A block diagram of our system architecture is depicted in Figure 2. Not discussed in detail in this paper are the pre-processing stage and the decision function. The reason for this is that we anticipate rather commonplace implementations of pre-processing and decision-making for our problem. Our data suggest a pre-processing stage consisting of a bank of bandpass filters (or short-term fft) augmented with some statistical features such as number of peaks per time unit. A simple maximum selector implements the class decision making.

4•0 CONCLUSIONS

We have discussed the problem of modelling long delays in neural networks for a particular classification problem. It was argued that the gamma memory structure offers a more flexible temporal representation than

feedforward tapped delay lines and leaky integrators. When the class cardinality is large, we recommend a separate network for each class and a (perceptron-like) decision-based training strategy.

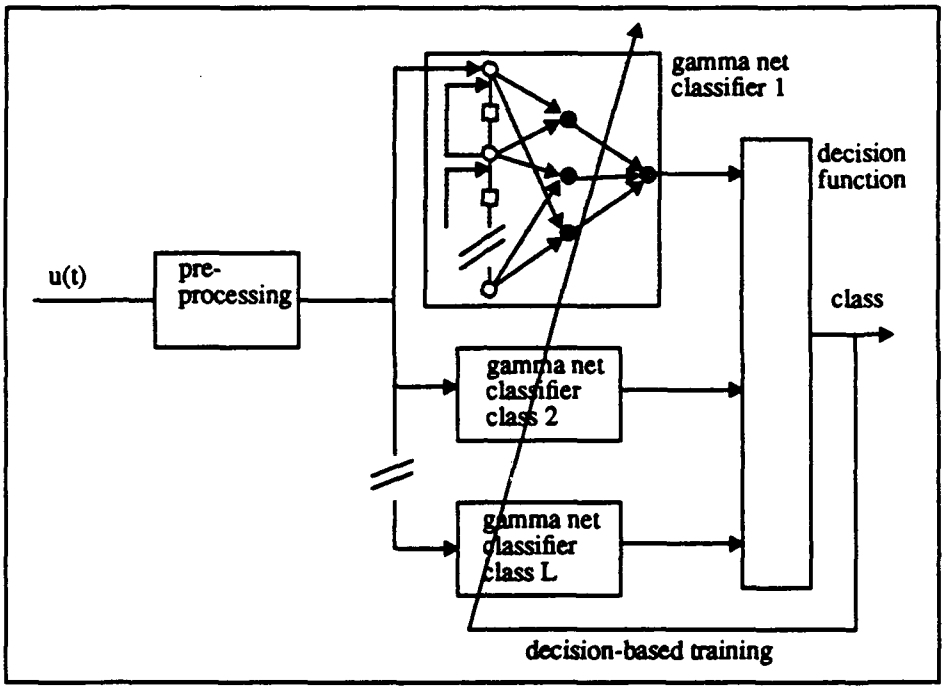


Figure 2. Block diagram of system architecture. Special features include gamma memory for retaining information over long delays, one network for each class, and decision-based training.

ACKNOWLEDGEMENTS

This work was performed under funding from DARPA and the National Information Display Laboratory.

REFERENCES

de Vries B., Temporal processing with neural networks: the development of the gamma model, Ph.D. dissertation, University of Florida, 1991.

de Vries B. and Principe J.C., A theory for neural networks with time delays, Advances in Neural Information Processing Systems 3, pp. 162-168, 1991.

Geman S., Bienenstock E. and Doursat R., Neural networks and the bias/variance dilemma, Neural Computation 4-1, pp. 1-59, 1992.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

NOVEL OPTICAL NEURAL NETWORK FOR TARGET DETECTION AND CLASSIFICATION

July 1992

J.N. Cederquist, D.K. Angell, J.H. Seldin
Optical and Infrared Science Laboratory
Advanced Concepts Division
Environmental Research Institute of Michigan
P.O. Box 134001, Ann Arbor, MI 48113-4001

K.A. Weigand
Air Force Wright Laboratory
Electronic Technology Laboratory WL/ELOT
Wright-Paterson AFB, Ohio 45433-5000

Abstract

A hybrid opto-electronic neural network using light emitting diodes, multimode waveguides, fixed pattern masks, and a linear photodetector is being analyzed, built, and demonstrated. The compact design, low power requirements, and high computational rate make this device attractive for a variety of target classification problems.

1.0 Motivation for Optical Neural Networks

Artificial neural networks (ANNs) are attractive for target classification of image data from visible, near infrared, multispectral, laser radar, synthetic aperture radar (SAR), or forward looking infrared (FLIR) sensors. Artificial neural networks can be used to preprocess scene data (e.g., edge enhancement) or segment an image (e.g., multispectral terrain classification). Preprocessed or segmented data from multiple sensors can be fused via an ANN to improve target recognition.

The high input data rates (10^8 data values/sec) are favorable to optical neural network processors. Optical artificial neural networks which perform pattern classification algorithms have undergone extensive theoretical and experimental analysis. These optical architectures include the Hopfield network, nearest neighbor networks, holographic architectures, and multi-layer feed forward architectures. To minimize physical requirements (size and weight) optical processing architectures have been developed which rely on solid-state and integrated optics technology. These architectures can be classified in terms of electronic[1] or optical[2] inputs with either electronic[2] or optical[1] weight masks. The weight masks can be variable or fixed depending on whether training is done on or off line based upon the application requirements. Systems that train on-line with real scene data require variable, programmable weight masks. The disadvantage of this approach is that increased power, packaging size, and dynamic range of weight values is required to adequately train the neural network.

We have developed an optical artificial neural network which provides high system throughput with minimal size, weight, and power. To demonstrate a near term ANN processor concept we are currently building a low power, optical-input, fixed optical weight artificial neural network scene segmentation processor for multispectral

classification.

2.0 Kohonen ANN for Multispectral Image Segmentation

The Kohonen artificial neural network[3] consists of a single input layer connected to a single output layer with each node having an associated neighborhood of other output nodes. For each input there is a single "winning" node which is the best match to the input vector. Unsupervised learning can be used to train the self-organizing Kohonen neural network. The Kohonen network is trained by presenting a member of the training set to the network and determining the "winning" node with minimum error between node weights w_{ij} and the input vector x_i (typical error metrics include Euclidean distance and inner products). The weights w_{ij} of all nodes in the neighborhood of the winning node are then updated and a new member of the training set is presented. Through successive iterations the neighborhood and the change in node weights are reduced and the system performance improves until the required level of system performance is achieved. The node weights specify clusters that reflect the probability density function of the input training vectors.

The Kohonen artificial neural network can successfully segment an agricultural image from a multispectral sensor[4]. We chose to use an optical neural network processor to implement a Kohonen network and process multispectral data for image segmentation. We have performed extensive studies to gain a better understanding of the numerical precision required of a hardware implementation of a neural network multispectral classifier. We concluded that adequate precision in the node weights is most important for reliable performance, and for this problem 6 bits was the recommended minimum precision. The necessary precision is most likely highly dependent on the particular neural network application and similar experiments must be performed prior to any specific hardware design. Our optical processor was designed to achieve this precision. A 5 x 5 Kohonen ANN prototype is currently being fabricated.

3.0 Optical Device Design and Fabrication

The optical ANN performs a vector-matrix multiplication of the input vector with the weight matrix. A schematic diagram of the optical ANN based on a single layer Kohonen self organizing neural network is shown in Fig. 1. Device construction is based on microfabrication technology. The data is input by LEDs into multiple multimode waveguides (rows in Fig. 1). The weight masks are area-modulated coupling of light out of the waveguides. This light is then summed by a linear array of photodetectors (columns in Fig. 1) to give the output. For increased reliability with compact, low power operation, all training is performed pre-mission and the neural network weights values are fixed. A parallel addressed LED emitter array (100 data values) operating at a modest 1 MHz rate results in an input data rate of 10^8 data values/sec. The computation rate for a single layer Kohonen network with 100 weights is $1x10^{10}$ operations/sec (ops). The device volume is on the order of 0.5 cm^3 resulting in a computation rate/unit volume of $2x10^{10}$ ops/cm^3 and power dissipation of 0.5 W/cm^3.

To demonstrate the concept we relied on commercially available LED sources and detectors. The vector input is provided by an LED (ROHM JA3012CL-01) array derived from an LED print bar with an 84.6 micron pitch, operating at 0.66 microns, with an active area of 50x65 $microns^2$. Light is coupled through an erect imaging SOLFOC array into a multimode dielectric filled waveguide (50x65 $microns^2$) interconnection mechanism fabricated in intrinsic silicon. Light is coupled out of the guide through an area modulated weight matrix which has been etched into the dielectric filled guides. The etched area is proportional to the weight matrix and was determined off-line via the multispectral classification simulation. The weight matrix mask was fabricated to 8 bit accuracy using e-beam photolithographic techniques. A commercially available photodetector (Reticon EG&G TB series) with an element spacing of 50 microns and an aspect ratio of 50:1 is used to sum the output of the input vector with the weight matrix.

The ANN electronics provides interfaces between the control computer and the electro-optical components of the experimental optical ANN system. The commercially available LED print bar does not provide multi-level inputs. To demonstrate the ANN, a pulse width modulation circuit was designed which can provide 5 bit input at a data rate of 100 KHz. The LED bar is controlled entirely by the computer, and the ANN electronics provide only buffering of control signals. For the diode sensor array, the ANN electronics generate a time delay and a burst of

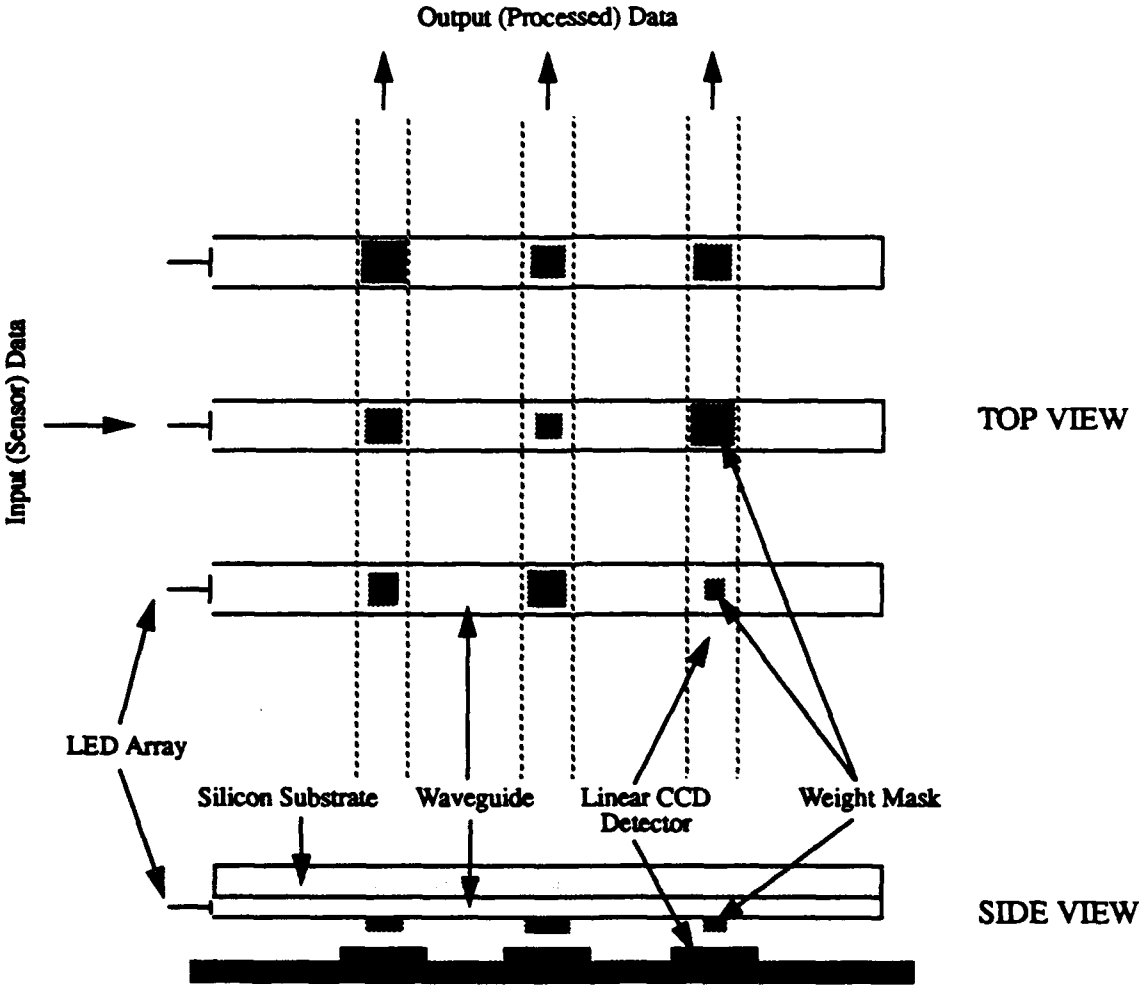


Figure 1. Optical artificial neural network (ANN).

sample pulses to drive the A/D converter. The A/D converter converts the analog video output of the sensor array to digital values to be processed by the computer.

The signal-to-noise (SNR) of the Reticon array output is a function of input power density at the photosite and the effective integration time. There are four primary noise sources to be considered for the TB series Reticon array. These are KTC noise (500 electrons at room temperature), amplifier read noise (100 electrons at room temperature), dark current noise, and signal shot noise. For a SNR of 3.5 the average input intensity is 1 nW/cm^2. The largest possible weight mask (40x10 $microns^2$) with an output efficiency of 0.1% results in an emission intensity of 0.664 $microwatts/cm^2$ and an output dynamic range of 664:1.

The fabrication of the waveguide array uses microfabrication technology. Thermal oxide (1.2 microns) is grown in intrinsic <110> silicon. The silicon is patterned with the waveguide mask to match the pitch and element spacing of the LED emitter array (84.5 micron pitch, 60% duty cycle). The oxide is etched with standard buffered oxide etch and the intrinsic silicon is etched to a depth of 65 microns with anisotropic wet etchant. Thermal oxide is then grown to a thickness of 1.2 microns on all etched surfaces forming an SiO_2 cladding layer of index 1.46.

Ultra-violet curing Norland optical adhesive of refractive index 1.56 is used to fill the guide and wafer polished flat to form a clean waveguiding structure. Positive photoresist is then used to pattern the weight mask.

We are currently fabricating the optical ANN structure for a multispectral segmentation algorithm. Experimental testing of the planar structure will be performed to determine channel cross-talk and dynamic range limitations of the weight mask. The ANN processor performance will be determined under a variety of input conditions. These results will be compared to digitally simulated results for a Kohonen multispectral segmentation application.

4.0 Summary

We have presented a new approach to optical artificial neural network processing based on waveguide optics technology which minimizes system size and power requirements while maximizing throughput. Our concept demonstration relies primarily on commercially available off-the-shelf hardware (LED sources, photodiode detectors) and easily fabricated multimode waveguide technology. We envision that this technology will provide a fault tolerant, rugged, compact approach to decrease the computational burden of digital processors for automatic target classification problems. This experimental approach is limited due to the serial nature of the LED source but provides a means to test the viability of the concept. With specialized parallel input/output hardware, operating at modest rates, computational rates of $2x10^{10}$ ops/cm^3 can be achieved dissipating 0.5 W/cm^3 of power.

5.0 Acknowledgments

We wish to express our thanks to B. Neagle and J. Abshier of ERIM and Jon Berrien and Sam Leforge of TSSI of WL/ELOT for their assistance in the development of the electronic control hardware We also wish to thank Russ Scherer of WL/ELRD for helpful suggestions on fabrication approaches and J. Marron of ERIM for his many helpful comments on the implementation of the optical neural module design.

This work was supported through funds provided by Wright Laboratories/ELOT (Contract Number F33615-90-1437) whose support is greatly acknowledged.

6.0 References

[1] E.A. Rietman, R.C. Frye, C.C. Wong, and C.D. Kornfeld, "Amorphous silicon photoconductive arrays for artificial neural networks," Appl. Opt. 28, 3474 (1989).

[2] J. Ohter, Y. Nitta, and K. Kyuma, "Dynamic optical neurochip using variable-sensitivity photodiodes," Opt. Lett. 16, 2342 (1991).

[3] R.P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, p. 4, April 1987.

[4] J.H. Seldin and J.N. Cederquist, "Classification of Multispectral Data: A Comparison Between Neural Network and Classical Techniques," Government Neural Network Applications Workshop, Dayton OH, 24-26 August 1992.

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

The SAHTIRN™ System for ATR (U)

William P. Lincoln, Cindy E. Daniell, Walter A. Tackett,
and Gregory A. Baraghimian

Hughes Aircraft Company, Missile Systems Group
8433 Fallbrook Avenue 262/C64
Canoga Park, CA 91304

(U) ABSTRACT

(U) We describe the SAHTIRN™ (Self Adaptive Hierarchical Target Identification and Recognition Neural Network) system. SAHTIRN™ combines three neural network models for automatic target recognition (ATR): (a) an early vision segmentor, (b) a hierarchical feature-extraction and aggregation recognition system based on the Neocognitron architecture, and (c) a multi-layer perceptron for pattern classification. Of these three models, the Neocognitron-based architecture has undergone extensive enhancements over previous literature reports based on unpublished in-depth analyses. We summarize our recognition system modifications and provide test results.

(U) The recognition core of the SAHTIRN™ system is a hierarchical, distortion-tolerant feature extraction and aggregation model. Feature extraction and aggregation are both self-organizing and undergo a training phase. Features important for target discrimination within expected scenes are learned during this training period. These primitive features are then aggregated by the hierarchical structure, forming a compressed representation of the input image. SAHTIRN™ learns to extract the spatial relationships of the primitive features, enabling identification of spatial structures of scenes and targets.

(U) An ATR system must have tolerance to such scene dynamics as clutter, occlusions and obscurations (bushes, trees, rock, hills, roads, dust, etc.), perspective changes, target signature and background thermal variations, target shift, rotation due to missile roll (both skid to turn and back to turn). Performance under each of these variations are further subject to differing detector resolutions and signal-to-noise ratios. We provide test results of the SAHTIRN™ ATR system against ground vehicular targets, systematically verifying the recognition abilities against the scene variations listed above. We used both computer and terrain board modeled infrared (IR) imagery in our evaluation, including imagery from DARPA's Artificial Neural Network Comparative Performance Assessment study.

(U) We discuss the Datacube hardware implementation of SAHTIRN™. Recent benchmarking indicates a four Hertz processing capability using a single i860 processor. Current efforts involve hardware development leading to a speed increase up to two orders of magnitude.

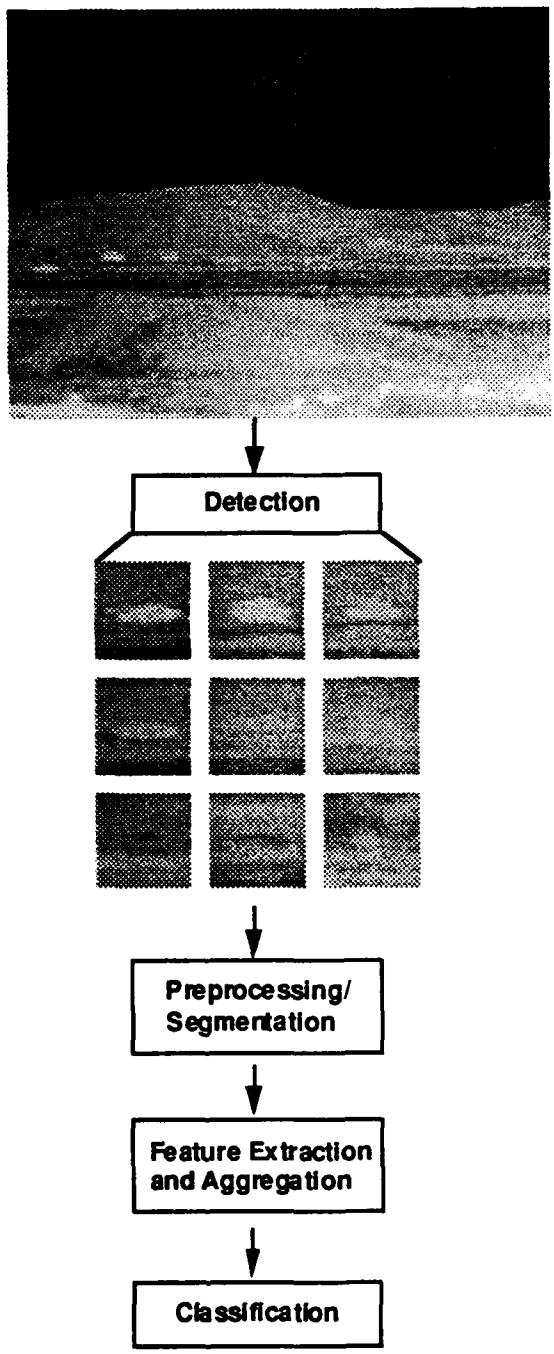
1.0 (U) INTRODUCTION

(U) We introduce SAHTIRN™ (Self-Adaptive Hierarchical Target Identification and Recognition Neural Network), a new neural network architecture resulting from unpublished theoretical "neocognitron"[1] analysis. The neocognitron is a hierarchical multilayered neural network with deformation-tolerant pattern recognition potential. The neocognitron has received much attention.[2,3,4]

(U) We summarize features of our neural network as a result of in-depth analysis of the neocognitron and its unsupervised learning algorithm. The breadth of the analysis spans clustering methods, signal processing, hyper-dimensional geometry and information theory. The motivation for this research is to realize the potential of a hierarchical multilayered deformation-tolerant family of neural networks.

(U) Based on analysis, we provide modifications, generalizations, and improvements to the neocognitron for elimination of ad-hoc parameter settings. In our new neural network, parameter settings are learned from the input data.

(U) We implemented the SAHTIRN™ system in realtime DATACUBE compatible hardware for field test preparation. Test results of this new architecture for automatic target recognition (ATR) using infrared (IR) imagery are given. Figure 1 shows the ATR application at a high level. This paper focuses on processes within the feature extraction and aggregation component of Figure 1. See our earlier paper for more information on the remaining components.[5]



(UNCLASSIFIED)

Figure 1. (U) The SAHTIRN™ ATR consists of four parts: target detection, preprocessing and segmentation of regions of interest, hierarchical object pattern recognition, and classification.

---

1 (U) Fukushima, K., "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, Vol. 15, No. 6, pp. 445-469, 1982.

2 (U) Barnard, E. and Casasent, D., "Shift Invariance and the Neocognitron," *Neural Networks*, Vol. 3, No. 4, pp. 403-410, 1990.

3 (U) Johnson, K., Daniell, C., and Burman, J., "Feature extraction in the neocognitron, " *IEEE International Conference on Neural Nets*, San Diego, CA, 1988.

4 (U) Menon, M. and Finnemann, K., "Classification of patterns using a self-organizing neural network," *Neural Networks*, Vol. 1, pp. 201-215, 1988.

5 (U) Daniell, C. E., Kemsley, D. H., and Bouyssounouse, X., "Comparitive evaluation of neural based versus conventional segmentors," SPIE Automatic Object Recognition, Vol. 1471, 1991.

2.0 (U) NEOCOGNITRON REVIEW

(U) Here, relevant aspects of the neocognitron are briefly reviewed. For a complete description of the neocognitron, see the work of Fukushima.[6]

(U) As shown in Figure 2, the neocognitron is a hierarchical network. The initial stage of the network is the input layer called UC_0, and consists of a two-dimensional array of pixels. After the input layer, the neocognitron consists of a number, l, of pairs of layers each consisting of one US-layer and one UC-layer. Each layer contains of a number of planes, K_l, consisting of a two-dimensional array of S-cells or C-cells. The S-cells receive their input from the C-cells in the preceding layer. The C-cells receive their input from the S-cells in the same layer.

(U) The purpose of each S-layer is to detect the presence of a particular pattern of activity in the preceding C-layer. This is enabled by a specialized two dimensional mask associated with each S-plane, called an a-mask. Each S-cell in a given plane is the result of correlation by the same a-mask but centered at differing positions in the preceding C-layer.

3.0 (U) NEOCOGNITRON ANALYSIS RESULTS

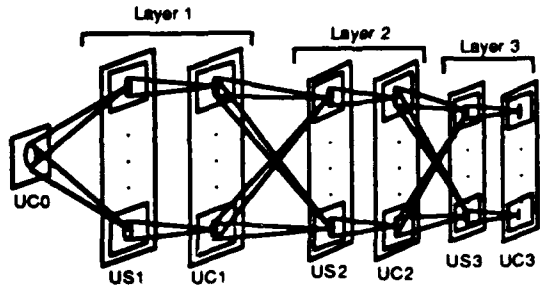
(U) Our neocognitron analysis leads to enhancements of Fukushima's original architecture.[7] We discuss five findings as a result.

(U) Detection Tolerance Mismatch

(U) The feature detection tolerance becomes mismatched with feature variance during learning. The situation is illustrated in Figure 3. Our unpublished analysis proves the neocognitron adapts the sizes of the feature decision boundaries *in the wrong directions* by becoming more tolerant to low variant features less tolerant to high variant features.

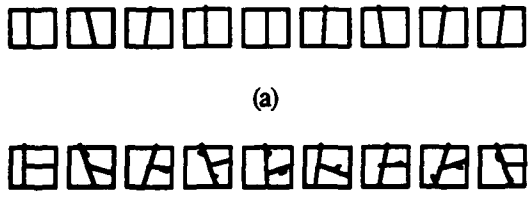
(U) Imprinting

(U) Imprinting is the indelible effect the first training patterns have on the correlation masks. This is clearly harmful during unsupervised learning. Due to the increasing



(UNCLASSIFIED)

Figure 2. (U) The hierarchical structure of the neocognitron. Planes in higher layers extract more complex features by hierarchically aggregating features from lower layers. Each plane learns to extract a specific feature.



(a)

(b)

(UNCLASSIFIED)

Figure 3. (U) Representative training receptive field patterns form the correlation masks for two different features. In (a) there is low feature variance, but the feature detection will become more tolerant during training. Conversely, in (b) there is high feature variance, but the feature detection will become less tolerant during training.

norm of the correlation masks during training, imprinting is observed in the neocognitron. This phenomenon has sometimes been called the Pinnocchio effect. We discuss solutions to imprinting below.

(U) Learning Based on Frequency of Occurrence

(U) Since the maximum responding S-cell is used to update a feature, the neocognitron learns the features in the training images that occur most often. These features may or may not be the most discriminating. It is often the case that a discriminating feature for a particular class will only occur once for each image in the class and will not be present in any of the images from the other classes.

---

6 (U) Fukushima, K., "Analysis of the process of visual pattern recognition by the neocognitron," *Neural Networks*, Vol. 2, pp. 413-420, 1989.

7 (U) Hecht-Nielson, R., Neurocomputing, Addison-Wesley Publishing Company, Redwood City, 1991.

(U) Uncorrelated Assumption

(U) The c-mask weights the pixels during the feature detection process. Traditionally the values of the c-mask are set in an ad-hoc fashion such that they decrease monotonically from the center of the c-mask. This ad-hoc setting is weak for two reasons:

a) (U) by using the c-mask, an implicit assumption is made that the pixels in a feature are uncorrelated with one another, and

b) (U) the c-mask values are preset and are not a function of the training images.

(U) Ad-hoc Parameter Choices

(U) Traditionally, a host of parameters must be chosen to use the neocognitron for a particular pattern recognition problem; such as, the number of layers, number of planes per layer, stopping criteria during training, decimation rates, mask sizes, c-mask and d-mask settings, and r parameter values.

(U) We summarize enhancements that eliminate these ad-hoc choices by learning the values of the parameters from the training images.

4.0 (U) MODIFICATIONS LEADING TO A NEW NEURAL NETWORK

(U) We present extensions to the neocognitron architecture based on the findings in the preceding section. These modifications fall in two categories.

(U) The first category of changes are relatively minor, keeping the flavor of the neocognitron's unsupervised learning. These are:

a) (U) making the feature detection surfaces explicitly hyperconic, and

b) (U) using unsupervised competitive clustering methods to solve the imprinting problem.

(U) The second category of changes, on the other hand, makes a break with the traditional neocognitron architecture and unsupervised learning. The result is a new neural architecture. The features of this are highlighted here. In summary, the new neural network and supervised batch learning algorithm optimize the feature detection and aggregation processes enabling the potential for deformation-tolerant pattern recognition.

(U) The new neural network presented in this section differs from the neocognitron the following ways:

a) (U) Avoids imprinting, the indelible effect of the first training patterns, during unsupervised learning.

b) (U) Learns discriminating features rather than frequently occurring features by maximizing a mutual information-theoretic measure between features and classes.

c) (U) Generalizes c-mask weighting to C-matrix weighting to obtain correctly shaped and oriented feature detection surfaces.

Traditionally, the c-masks were set ad-hoc.

d) (U) Computes the r parameter from the training images. The r parameter determines the angle of aperture of the hyperconic decision surface cone. Traditionally, the r parameters were set ad-hoc.

e) (U) Determines the number of feature detectors in each layer during learning based on the number of discriminating features learned from the training data.

f) (U) Feature learning asymptotes to useful, discriminating features rather than requiring an ad-hoc stopping criteria for learning.

g) (U) Estimates an r parameter and C-matrix for each feature detector plane instead of one r parameter and c-mask per layer.

h) (U) Estimates the scale and distortion allowed in the aggregation process (i.e., learns the d-masks, number of layers, and eliminates the need for decimation).

5.0 (U) RESULTS AND HARDWARE IMPLEMENTATION

(U) Recognition results based on our developments are shown for an ATR application using simulated infrared (IR) imagery. Results obtained use a fully automatic preprocessing system. We performed two experiments, testing SAHTIRN™'s ability to discriminate between three different vehicles (e.g., tanks, trucks, helicopters).

(U) In the first test with vehicles at a range of 4.5 km, we used 30 training and 30 test regions of interest. Results are 90% correct classified, 10% misclassified, and 0% rejected.

(U) The second test used 66 images of vehicles (at ranges from 3.5 to 4.5 km) and additional clutter (i.e., no vehicle present) images for training. Performance results for 66 test vehicle images are 92.4% correct classification, 4.6% misclassification, and 3% rejected.

(U) The SAHTIRN™ system is implemented in realtime DATACUBE hardware and prepared for field testing.

6.0 (U) CONCLUSION

(U) We presented a new neocognitron-based neural network architecture that robustly recognizes patterns. The goal of our research is to develop a theoretical understanding of the hierarchical feature extraction and aggregation capabilities of this neural architecture. We provided recognition results of this new architecture for ATR using IR imagery.

(UNCLASSIFIED)

Figure 4. (U) Our SAHTIRN™ Datacube implementation currently runs at 4 Hz on one i860 processor. Shown here are: 1) a SUN controller workstation monitor (lower left), 2) the Datacube chassis (lower right), and 3) a video monitor for I/O display (top).

(U) ACKNOWLEDGMENTS

(U) The authors wish to thank everyone in the Imaging Guidance Design Laboratory of Hughes Aircraft Company, Missile Systems Group, and Dr. Kenneth Friedenthal for continuing support in this effort.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

Neural Networks for Classifying Image Textures

Stephen H. Lane, John C. Pearson, and Ronald Sverdlove

National Information Display Laboratory
at the
David Sarnoff Research Center
CN 5300
Princeton, New Jersey 08540
609 734-2517

ABSTRACT

It is shown that neural networks can be used effectively to classify areas of a satellite image of the earth according to image textures such as forests, fields, and buildings. Such classification can be useful in the exploitation of large image databases. The Spline-Net neural network architecture was found to provide an efficient solution to this problem using a modest amount of training data.

INTRODUCTION

In the complex world of interactive image and signal exploitation, large amounts of data arrive every day, much of it requiring immediate attention. It is the analyst's job to scan this information and determine its meaning and significance. In order to assist the analyst with the complex and time-consuming task of exploiting image databases, computer-based tools are needed that can search images for relevant features and content.

In many applications, one is looking for those images from a large set which contain particular objects. To simplify the process, it may be useful to extract a subset of the image database which is relevant to the target being sought. For example, if one is looking for ships, only those images containing bodies of water need to be looked at. If airplanes are the targets, images containing large paved areas (runways) would be the relevant subset. Since a body of water or a paved area that can be used as a runway does not have a well-defined shape, one cannot use standard pattern matching techniques to find it. However, those areas in an image do have certain visual "textures" that can be used to distinguish them from surrounding land. Textures are also important in themselves for studying environmental features such as forests or fields of crops.

IMAGE TEXTURE CLASSIFICATION

Visual texture is produced by repetitions of small-scale elements or features in a scene. By "small-scale" we mean small enough that they cannot easily be perceived as individual objects. The statistical distributions of these elements or features may help to distinguish one texture from another. A region of one texture has a certain uniformity of appearance. Where two distinct texture regions meet, there appears to be a sharp boundary curve, even though the physical change may be gradual. Perceived texture may be independent of viewing distance, because the elements occur at many different scales. For example, a forest viewed from two different altitudes is easily recognizable as such. However, when one gets so close to a texture region that one can see the individual elements (e.g. trees) the forest texture is no longer seen.

A person can look at an image of an area on the earth and quickly and easily distinguish between forests, fields, bodies of water, towns, etc. Many models have been proposed for how the human brain perceives texture [1]. Machine vision systems have been developed which can find the boundaries between texture regions in some cases. However, for applicability to a wide variety of problems, it is useful to have a system which can learn from examples without being dependent on a particular model. Neural networks are well suited for this sort of pattern classification. Detailed knowledge of the process which generated the training data is not required for the network to do a good job. Neural networks are also appropriate since they can divide an input space into disjoint regions with nonlinear boundaries and can run very quickly when implemented in computer hardware.

To construct a network for texture classification, we must first preprocess the image to reduce the amount of data. If every pixel in a 1000×1000 image were used as an input to a network, the size of the network would make training impractical. The first step is to divide the image up into smaller blocks, each of which is to be assigned to one of a fixed set of texture classes, or to none of them. The network architecture will have one output node for each of the classes. A block will be assigned to a class if the corresponding output node has a high response to the input data respresenting the block while all the other output nodes have a low response. The pixel values in the block must then be processed to produce a small number of inputs for the network. A good method of doing this involves the use of "oriented filter pyramids." A multiresolution representation of the data is created by passing it through a series of two-dimensional linear bandpass filters with different orientations and scales. Each filter acts as a feature detector on the whole block. These filters can be computed very efficiently using the pyramid algorithms of Burt [2]. Bergen [1] explains why this process preserves the significant information required for texture perception. The energy in the output of each filter becomes one input for the neural network classifier. A small number of these energy measures can result in good classification with a simple network that can be trained quickly.

SPLINE NETS

Spline-Nets [5,6] are part of a larger effort to develop neural network architectures and training methods tailored specifically for image processing and signal exploitation systems [3-4]. Spline-Nets are generalized versions of Multi-Layer Perceptrons (MLP) that incorporate B-Spline connection functions into the network node computations in an attempt to trade-off additional hidden layers for node complexity. The overall effect is to combine the fast learning and computational efficiency of strictly local network architectures with the scaling and generalization properties of standard MLPs.

In a standard MLP, shown in Fig. 1, the connection functions that relate the output or activation of one node to the input of another is simply a linear function with the slope equal to the value of the weight.

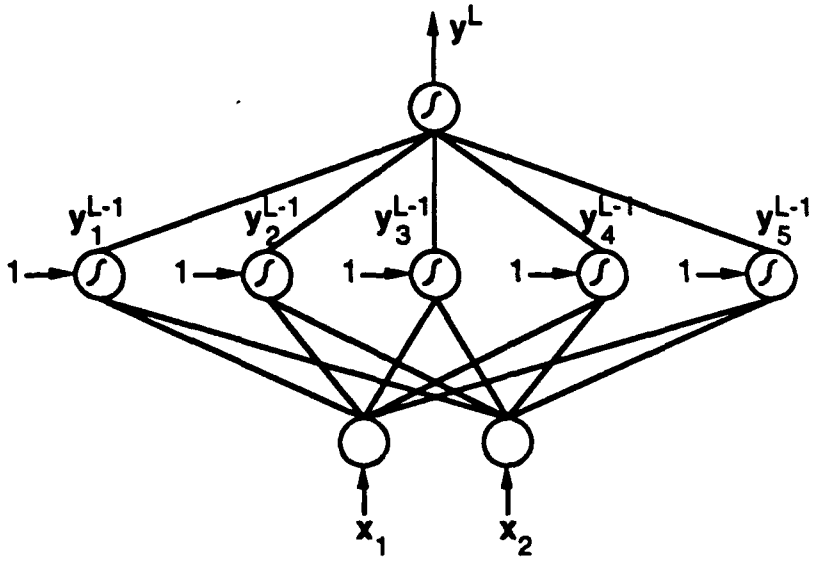


Figure 1. Standard Multi-Layer Perceptron

Standard Multi-Layer Perceptrons (MLPs) can be represented using node equations of the form

$$y_i^L = \sigma\left(\sum_{j=0}^{\eta_{L-1}} c_{ij}^L\right) = \frac{1}{1+\exp\left(-\sum_{j=0}^{\eta_{L-1}} c_{ij}^L\right)} \tag{1}$$

where h_L is the number of nodes in layer L and the c_{ij}^L are linear connection functions between nodes in layers L and $(L\text{-}1)$ such that,

$$c_{ij}^L = w_{ij}^L y_j^{L-1} \tag{2}$$

$\sigma(\bullet)$ is the standard sigmoidal nonlinearity, y_i^{L-1} is the output of a node in layer L-1, $y_0^{L-1} = 1$, and the w_{ij}^L are adjustable network weights. Some typical linear connection functions are shown in Fig. 2. c_{D}^L corresponds to a threshold input.

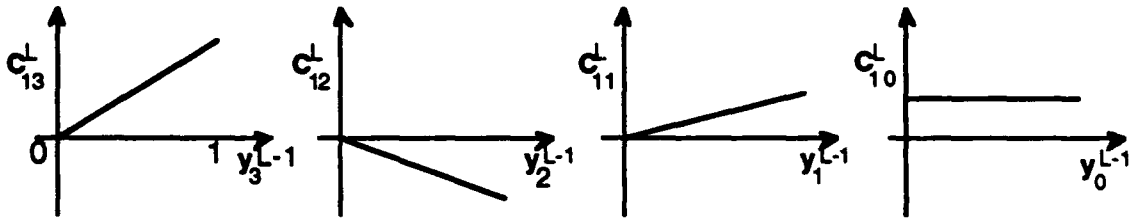


Figure 2. Typical MLP Node Connection Functions

Spline-Nets generalize this concept of connection functions by representing them using splines (piecewise polynomials). This is allows more effective connection functions (e.g. piecewise linear, quadratic, cubic, etc.) to be formulated. The corresponding B-Spline MLP (Spline-Net) is derived by redefining the connection functions of eq. (2) such that,

$$c_{ij}^L(y_j^{L-1}) = \sum_k w_{ijk}^L B_{nk}^G(y_j^{L-1}) \tag{3}$$

This produces a more general neural network architecture (known as a Spline-Net) that has node equations of the form,

$$y_i^L = \sigma\left[\sum_{j=1}^{\eta_{L-1}} c_{ij}^L(y_j^{L-1})\right] = \frac{1}{1+\exp\left[-\sum_{j=1}^{\eta_{L-1}} c_{ij}^L(y_j^{L-1})\right]} \tag{4}$$

The $B_{nk}^G(y_j^{L-1})$ are B-spline receptive field functions [5,6] of order n and support G, while the w_{ijk}^L are the Spline-Net weights. The order, n, corresponds to the number of coefficients in the polynomial pieces. For example, linear splines are of order n=2, whereas cubic splines are of order n=4. The advantage of the more general B-Spline connection functions of eq. (3) is that it allows varying degrees of "locality" to be added to the network computations since network weights are now activated based on the value of y_j^{L-1}. The w_{ijk}^L are modified by backpropagating the output error only to the G weights in each connection function associated with active (i.e. nonzero) receptive field functions. The L^{th}-layer weights are updated using the method of steepest descent learning such that,

$$w_{ijk}^{L} = w_{ijk}^{L} + \beta e_i^{L} y_i^{L}(1 - y_i^{L}) B_{nk}^{G}(y_j^{L-1}) \tag{5}$$

where e_i^L is the output error back-propagated to the i^{th} node in layer L and b is the learning rate [5,6]. In the more general Spline-Net formulation of eqs. (3-5), each node input has $P+G-1$ receptive fields and $P+G-1$ weights associated with it, but only G are active at any one time. P determines the number of partitions in the input space of the connection functions. Standard MLP networks are a degenerate case of the Spline Net architecture, as they can be realized with B-Spline receptive field functions of order $n=2$, with $P=1$ and $G=2$. Due to the connectivity of the B-Spline receptive field functions, for the case when $P>1$, the resulting network architecture corresponds to multiply-connected MLPs, where any given MLP is active within only one hypercube in the input space, but has weights that are shared with MLPs on the neighboring hypercubes. The amount of computation required in each layer of a Spline Net during both learning and function approximation is proportional to G, and independent of P.

Formulating the connection functions of eq. (3) with linear ($n=2$) B-Splines allows connection functions such as those shown in Fig. 3 to be learned.

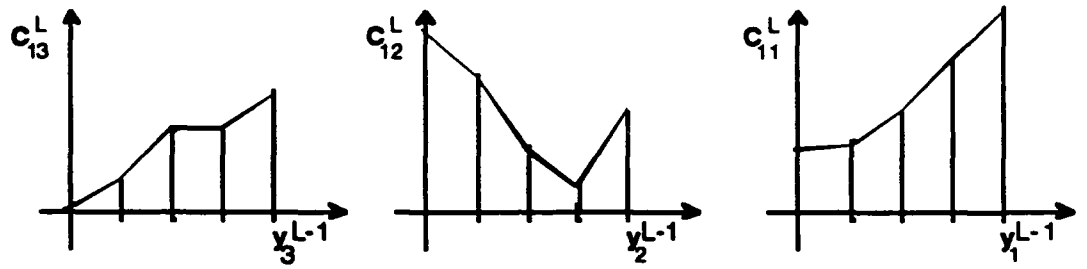


Figure 3. Spline Net Connection Functions Using Linear B-Splines ($n=2$)

The connection functions shown in Fig. 3 have $P=4$ partitions (5 knots) on the interval $y_j^{L-1} \in [0,1]$. The number of input partitions, P, determines the degree of locality of the resulting function approximation since the local shape of the connection function is determined from the current node input activation interval. For the linear splines shown in Fig. 3, each connection function partition has associated with it two network weights.

Initially the Spline-Net connection functions have two knots, one located at 0 and the other one at 1, making them equivalent to those found in a standard MLP. During training, additional knots and weights are incrementally introduced into the connection functions. This is done by using a bifurcation schedule [5,6] which splits each partition in half and reintializes the network weights such that the shape of the connection functions before and after the split look the same. The bifurcations occur when the slope of the mean-square-error curve drops below some threshold value. The overall result of this type of training is to carve-out coarse global features first, then capture finer and finer localized details later, thereby automatically matching the complexity of the network design to the complexity of the problem. This not only improves generalization, but reduces the amount of training required to obtain a given level of performance. Use of a bifurcation schedule also simplifies the user interface by making the number of nodes in the hidden layer the primary parameter specified to obtain an acceptable network design.

TEST PROBLEM

To test the approach described above, we used data from satellite images of the earth and sought to classify regions as having one of three types of texture: forests, fields, or buildings. To create a training set, areas were chosen by hand, from a number of images, that were in each of the three classes. For each class, a mosaic image was formed consisting entirely of examples of that class. Each training example was created by processing a 32 × 32 pixel block with the oriented filter pyramid as shown in Fig. 4.

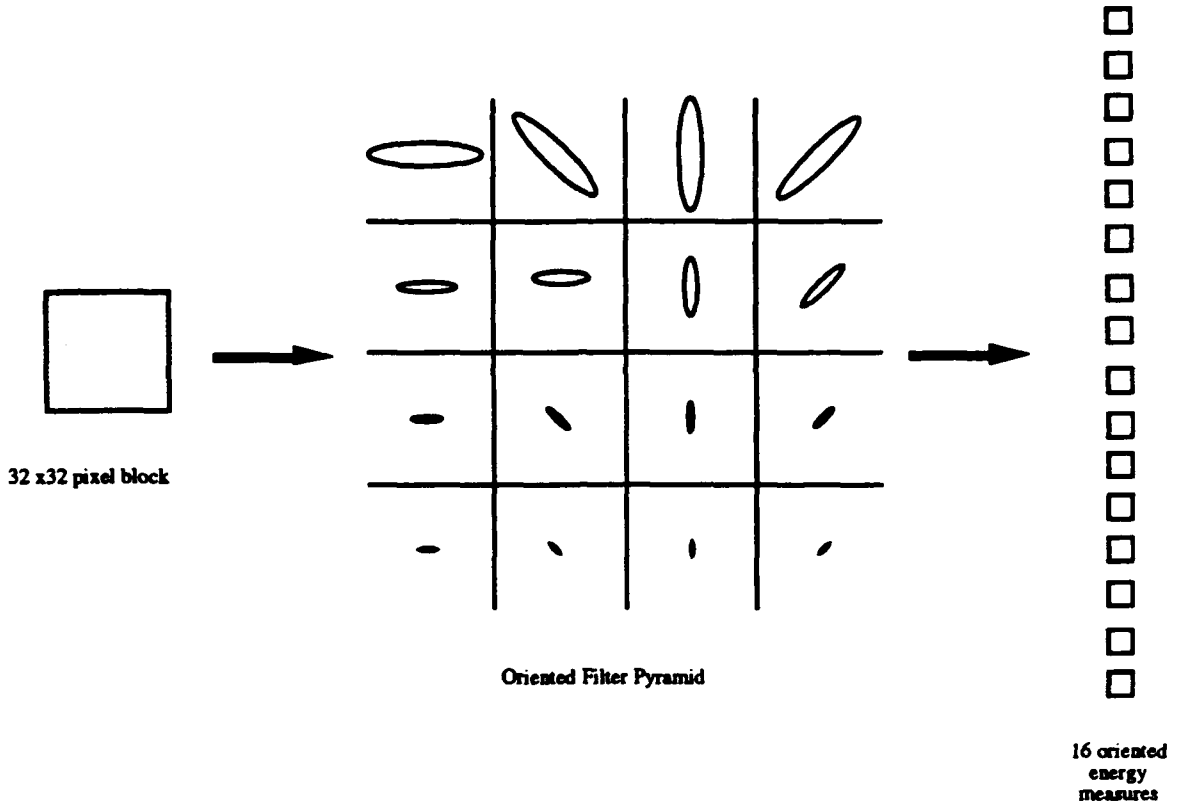


Figure 4. Preprocessing of image blocks to create network input.

The pyramid consisted of four scales, differing by a factor of two, and four orientations differing by 45°. Each of the three mosaic images was 512 × 512 pixels so there were a total of 256 training examples for each class.

TEXTURE CLASSIFICATION RESULTS

Classification of terrain textures similar to those found in the satellite photograph of Fig. 5 were conducted using the Spline-Net architecture shown in Fig. 6 having linear B-Spline connection functions with support G=2, one hidden layer containing 8 hidden nodes, and 3 output nodes. Each of the network output nodes corresponds to one of three textures in the satellite image: forests, fields, or buildings. Spline-Net training was accomplished using the "vanilla" back-prop learning rule of eq. (5) with $\beta = 1/(2P)$. The connection function weights were initialized in each node such that the resulting connection functions were continuous linear functions with arbitrary slope. During training, the connection functions were made piecewise linear by bifurcating all partitions (splitting them in half) after every 100 training epochs. An epoch for this problem consisted of a sequence of 768 training examples (256 for each texture) and the particular order of the exemplars in the training set was reshuffled after every epoch. The reason for gradually increasing the connection function input partitions during training is to improve the network generalization capability and increase the learning speed [5,6]. Although the effect of such partition splitting effectively doubles the total number of weights in the network, it has no effect on the actual number of weights used in computing a network output due to the finite support (nonzero over only 2 adjacent partitions) of the linear basis splines used.

The results presented in Figs. 7-11 were generated by training the network shown in Fig. 6 with a total of 700*768 examples. The training of the Spline-Net of Fig. 6 was begun by initializing it as a standard MLP with P=1. Figure 7 shows the percent of textures correctly classified by the network, and Fig. 8 shows the corresponding learning curve. The percent classified correctly is based on all three network output nodes in Fig. 6 having the proper values, (1 if a texture is present, 0 if it is not). A threshold value of 0.5 was used to decide whether a node

was outputting a 1 or a 0. The results associated with the first 100 training epochs in Figs. 7 and 8 correspond to what would be produced by a standard MLP classifier of similar topology. However, with a Spline-Net, each time the connection functions partitions are split by the bifurcation schedule, there is a significant drop in the mean-square-error of the network This can be clearly seen in Fig. 8 after every 100 training epochs. Once the connection functions have 16 partitions very little is gained by the additional training using 32 and 64 input partitions. In contrast, if the standard MLP (1 partition) of the first 100 training epochs was continued unchanged and trained on the entire data set (700*768 examples), the resulting mean-square-error would be about four times higher than the Spline-Net results shown in Fig. 8 and the percent classified correctly would be approximately 8% worse. Figure 9 shows the percentage of each texture correctly classified by its corresponding output node. Figures 10 and 11 are histograms that show how the distribution of values produced by each output node on the texture data set varies as the number of connection function partitions is increased from 1 to 16 during training. In these figures the output node values (<1 0 0>, <0 1 0 >, and <0 0 1>) produced by the 256 training examples corresponding to each texture are distributed into 100 bins between 0 and 1. The height in a particular bin is directly proportional to the number of training examples that produced an output value in that quantization interval. The results of Figs. 10 and 11 are useful when determining optimal threshold values for the output nodes (as opposed to 0.5) that maximize the number of correctly classified textures.

Figure 5. Typical terrain textures found in satellite images.

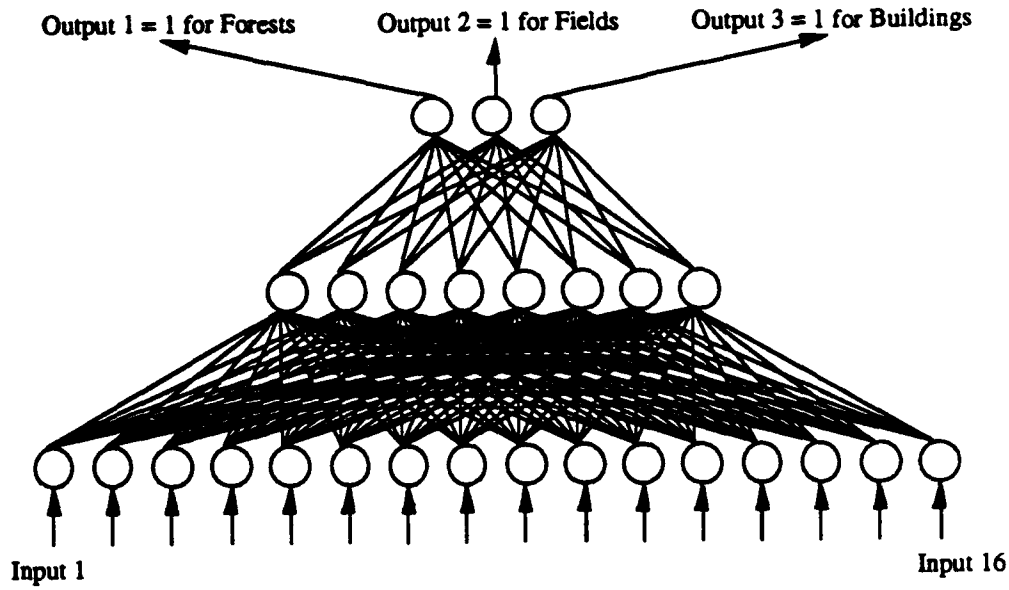


Figure 6. Three-Layer Spline-Net with 16 inputs, 8 hidden nodes, and 3 output nodes.

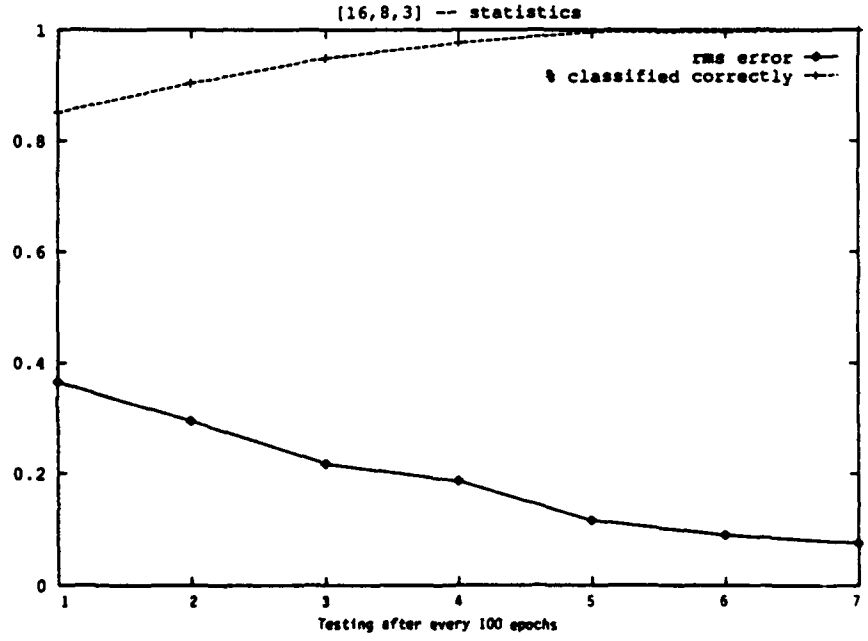


Figure 7. Percent Textures Classified Correctly by Network

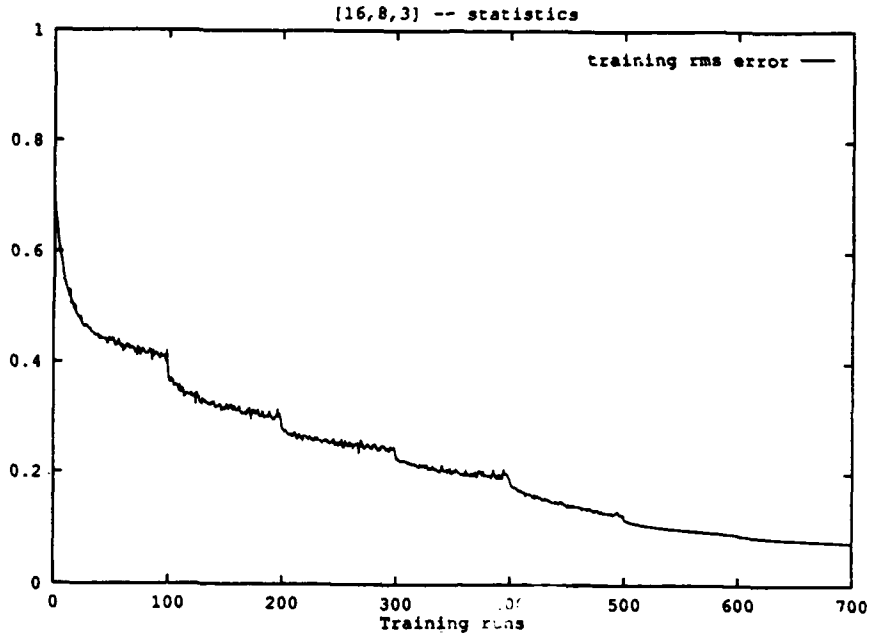


Figure 8. Root-Mean-Square (RMS) Error Learning Curve.

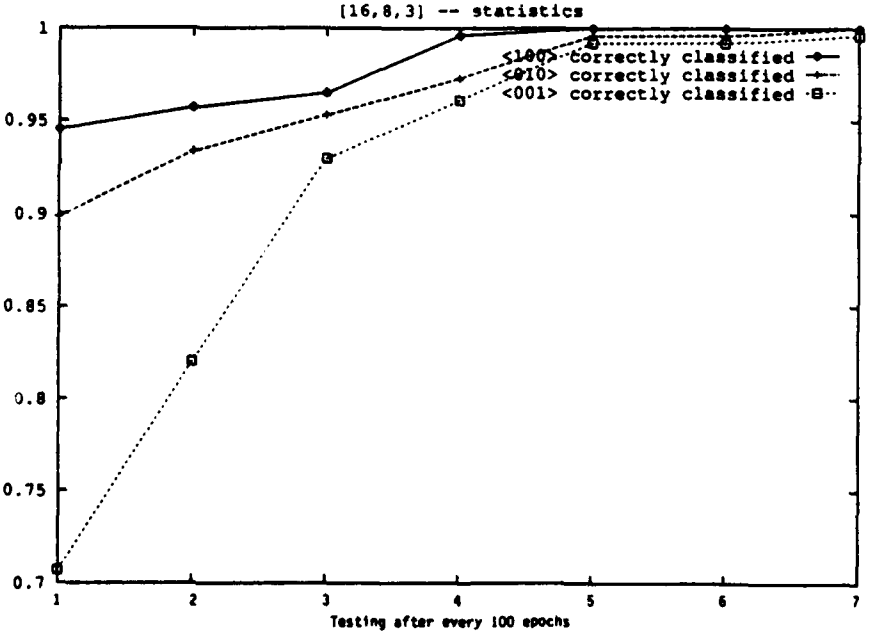


Figure 9. Percent Textures Classified Correctly by Individual Output Nodes

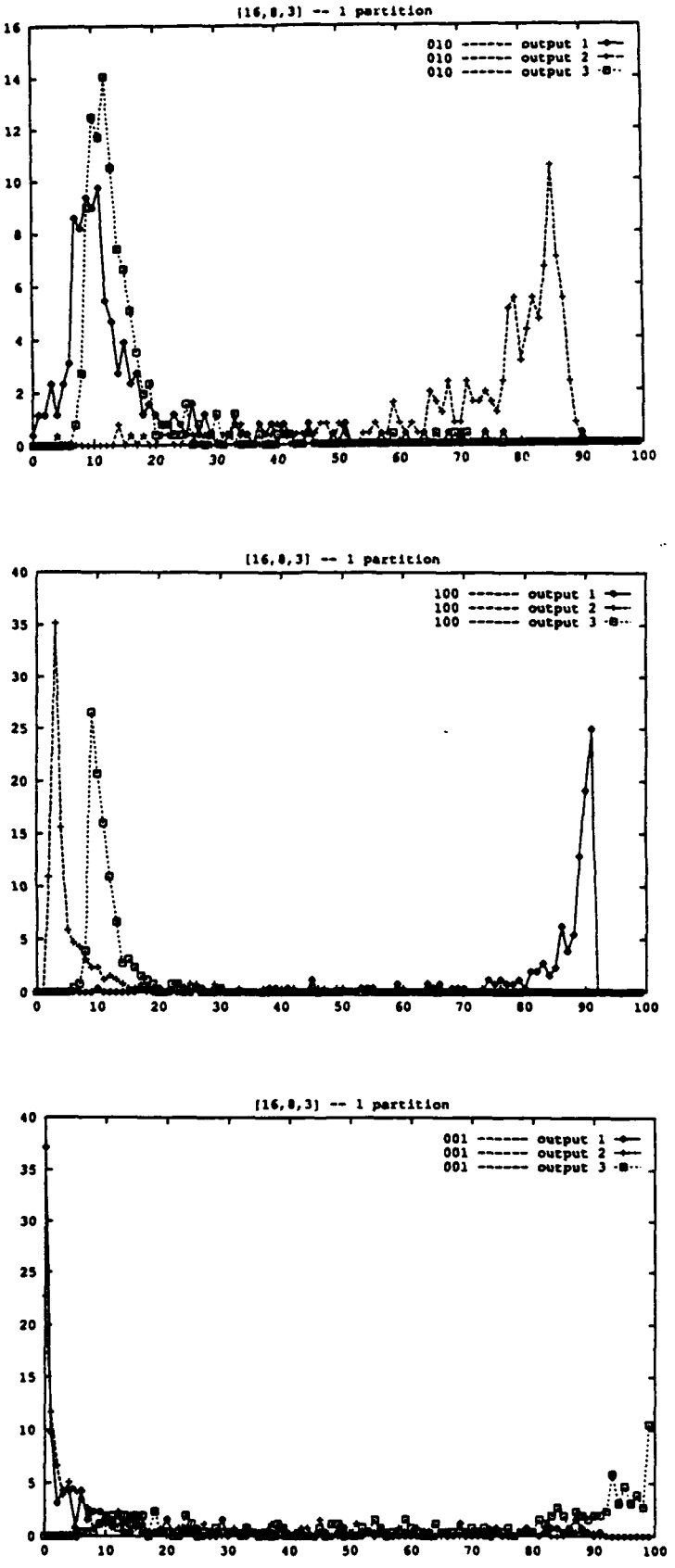


Figure 10. Distribution of Spline-Net Output Values on Texture Training Set when P=1.

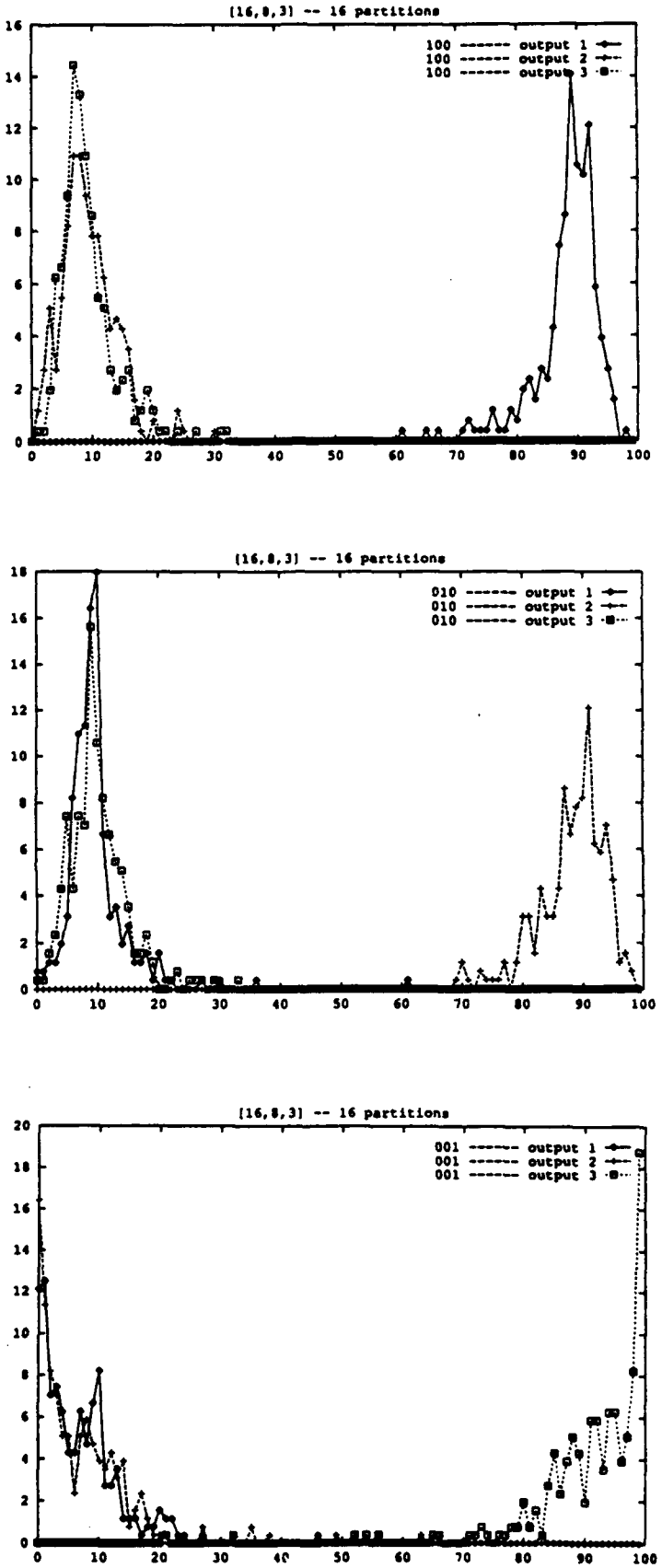


Figure 11. Distribution of Spline-Net Output Values on Texture Training Set when P=16.

CONCLUSIONS

It was shown that typical textures found in satellite imagery can be correctly classified by a neural network with input produced by preprocessing the image using multi-resolution oriented linear filters for feature extraction. Only a modest amount of training data, created by hand, was necessary for succesful training. The Spline-Net neural network architecture was found to be more effective than a standard MLP of similar topology for this application. The results indicate that many difficult texture classification problems may be solved faster and more efficiently using single hidden-layer Spline-Net architectures and a training bifurcation schedule to determine the ideal number of Spline-Net connection function partitions.

REFERENCES

[1] Bergen, J.R. (1989). "Theories of Visual Texture Perception," **Vision and Visual Dysfunction, v. 10b**, D. M Regan, Ed., Macmillan.

[2] Burt, P.J. (1988). "Smart Sensing with a Pyramid Vision Machine," **Proceedings of the IEEE**, vol. 76, pp. 1006-1015.

[3] Pearson, J.C., Spence, C. and Sverdlove, R. (1991). "Application of Neural Networks in Video Signal Processing,"**Advances in Neural Information Processing - 3**, R.P Lippmann, J. Moody, D. Touretzky, Eds., Morgan Kaufmann.

[4] Binenbaum, N., Dias, L., Hsieh, P., Ju, C.H., Markel, S. Pearson, J.C. and Taylor Jr., T., (1991). "Neural Networks for Signal/Image Processing Using the Princeton Engine Multi-Processor," **Neural Networks for Signal Processing**, B.H. Juang, S.Y. Kung and C.A. Kamm, Eds., IEEE Press, pp. 595-605.

[5] Lane, S.H., Flax, M.B., Handelman, D.A. and Gelfand, J.J. (1991). "Multi-Layer Perceptrons with B-Spline Receptive Field Functions," **Advances in Neural Information Processing - 3**, R.P Lippmann, J. Moody, D. Touretzky, Eds., Morgan Kaufmann.

[6] Lane, S.H., Flax, M.B., Handelman, D.A. and Gelfand, J.J. (1991a) "Function Approximation in Multi-Layer Neural Networks with B-Spline Receptive Field Functions," **Princeton University Cognitive Science Lab Report No. 47.**

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

RETRIEVING ATMOSPHERIC TEMPERATURE AND MOISTURE PROFILES FROM DMSP SOUNDER DATA WITH A NEURAL NETWORK

C.T. Butler and R.v.Z. Meredith[1]
Physical Sciences Inc.
635 Slaters Lane, Suite G101
Alexandria, VA 22314

ABSTRACT

In recently completed studies, we accurately retrieved atmospheric temperature and moisture vertical profiles from simulated millimeter-wave sounder data. A similar study using operational data is currently getting underway. In the simulation studies, backpropagation networks were trained on data that simulated the multichannel output of either the SSM/T-1 temperature- or SSM/T-2 moisture-sensing radiometers on current Air Force DMSP weather satellites. Ground-truth information was obtained from a collection of radio- and rocketsonde measurements. Radiometer outputs were simulated from these data using the Air Force program RADTRAN, and training/testing sets constructed. For both temperature and moisture, the neural method produces atmospheric profiles from simulated, unfamiliar data that are comparable to or better than those obtained with current operational methods. Networks, however, were able to retrieve profiles from the simulated data over a much broader range of geographic areas and seasons than standard methods, and to do this using less externally supplied geographic or season information. Preliminary results using operational data will be given if available.

1. INTRODUCTION

The DoD relies on the SSM/T-1 and SSM/T-2 passive radiometers (sounders) of its DMSP weather satellites for vertical profiles of atmospheric temperature and water vapor, respectively. As the satellites pass over Earth's surface, the sounders continually measure the power received by their several channels. These power measurements are ultimately reported as effective, or "brightness" temperatures. Because they lie in different portions of atmospheric absorption bands, the channels each sense the brightness temperature at a different level of the atmosphere. A channel near the peak of the absorption band measures its parameter (temperature or moisture) at high altitudes, since little energy can penetrate to the sensor from lower levels; the farther off-peak the channel is, the lower the level of the atmosphere it samples. Arranged in the proper order, the channels of an instrument thus represent a rough ground-up profile of its measured parameter.

To obtain vertical temperature or moisture profiles, these brightness temperatures are currently fitted to solutions of the radiative-transfer equation that physically models the transmission of thermal and other radiation by the atmosphere,[2,3] or they are related to actual profiles by a piecewise-linear statistical-correlation method that "retrieves"

---

[1] Research Support Instruments, Inc., Alexandria, VA.

[2] Schaerer G., and T.T. Wilheit, " A Passive Microwave Technique for Profiling of Atmospheric Water Vapor," Radio Science, 14, pp. 371-375.

a plausible vertical temperature or moisture profile based on a training set composed of similar brightness temperatures and corresponding measured profiles or "ground truth."[4]

A statistical retrieval method based on a neural network should have two advantages over the one currently used. First, the natural ability of networks to model nonlinear phenomena should allow them to build a more accurate model of the highly nonlinear training data. For a given retrieval accuracy, this ability also should allow a network to work over a broader range of geographic area, terrain types, and seasons than the standard statistical method. Second, the ability of neural networks to fuse the output of more than one sensor will allow them to simultaneously retrieve several atmospheric parameters. This potentially allows more accurate retrieval of each parameter, since information from all sounder channels is available to the network as it builds its internal model of the training data.

2. SIMULATION STUDIES

The first of the two simulation studies reported here determined the ability of a neural network to retrieve temperature vertical profiles of the atmosphere from simulated DMSP SSM/T-1 (T-1) sounder data; the second determined the ability of a network to retrieve moisture vertical profiles from simulated SSM/T-2 (T-2) data. Both studies used fully connected, feed-forward networks and the backpropagation training algorithm. The operational study is briefly described in Section 3.

2.1. Data Preparation

Simulated brightness temperatures for both studies were generated by applying the program RADTRAN to measured atmospheric temperature or moisture profiles.[5],[6] RADTRAN uses a physical model of the atmosphere to compute, for a supplied profile, the brightness temperatures at the top of the atmosphere in selected wavelength bands. The T-1 instrument senses seven bands in the range from approximately 50-60 GHz; the T-2 instrument effectively senses five bands in the range from 90 to 190 GHz. The measured atmospheric profiles were taken from the so-called Phillips data, a set of 1600 temperature and humidity profiles assembled and standardized from radio- and rocketsonde measurements made at continental and maritime weather stations at locations from 30°S to 60°N latitude during two short periods in Northern-Hemisphere winter and early summer.[7]

Details of the data treatment differ between the two studies. In the temperature study, only over-land data were used and southern semitropical data were excluded, which resulted in a training set of 347 and a testing set of 328 examples. To allow unambiguous interpretation of the retrieval results, the emissivity of Earth was set to a constant value of 0.7 characteristic of land, and zero atmospheric moisture content was assumed. Thus the training and testing sets consisted of a mixture of atmospheric-profile/brightness-temperature pairs obtained over land, during both seasons, and from 10°S to 60°N latitude. The long training times experienced with the 20 MHz PC used for this portion of the study precluded using resampling methods that would have afforded larger training and testing sets.

---

[3] Kakar, R.K., "Retrieval of Clear Sky Moisture Profiles Using the 183 GHz Water Vapor Line," J. of Climate and Applied Meteorology, 23, pp. 1282-1289 (1983).

[4] Kakar, R.K., and Lambrigtsen, B.H., "A Statistical Correlation Method for Retrieval of Atmospheric Moisture Profiles by Microwave Radiometry," J. of Climate and Applied Meteorology, 23, pp. 1110-1114 (1984).

[5] Falcone, V.J., L.W. Abreu, and E.P. Shettle, "Atmospheric Attenuation of Millimeter and Submillimeter Waves: Models and Computer Code," Air Force Geophysical Laboratory, Hanscomb AFB, MA Report AFGL-TR-79-0253 (1979).

[6] Falcone, V.J., L.W. Abreu, and E.P. Shettle, "Atmospheric Attenuation in the 30-300 GHz Region Using RADTRAN and MWTRAN," Proceedings of the Society of Photographi and Optical Instrumentation Engineering, 337, 62 (1982).

[7] Phillips, N., Suskind, J., and McMillin, L., "Results of Joint NOAA/NASA Sounder Simulation Study," Journal of Atmospheric and Oceanic Technology, 5, 44 (1988).

The moisture study was carried out on a Sun SPARCstation 2, and a resampling training/testing method, four-fold cross-validation, was employed.[8] For this study, only over-ocean data were used, which led to a 600-member training set. Surface emissivity was again set to a constant value, this time to 0.4.

For each study, training and testing files were created with input vectors consisting of the five or seven sounder-channel brightness temperatures and any selection of available geographic and season cues. Latitude, solar zenith angle, and hour of observation were the only explicit cues available, but terrain-height and ocean/land cues were implicit in the training/testing data because of the structure of the Phillips data and choices made for each study. For moisture retrievals using any method, it is necessary to provide the temperature profile corresponding to the moisture data. To keep interpretation of results straight forward, we supplied these profiles from the Phillips data instead of co-retrieving them from the brightness temperatures computed for the temperature study. Desired outputs for the training files were the values of temperature or moisture at selected pressures starting at 1000 mb.

2.2. Network Training and Testing

2.2.1. Temperature. The backpropagation networks were implemented in simulation on a 386/387 PC using the commercial package ANSim. Four-layer networks were investigated, but were not superior to smaller ones; all results reported are from networks with a single hidden layer. Uncued networks contained seven and cued networks eight to ten input neurodes. The output layer consisted of up to 32 neurodes representing the atmospheric temperature at logarithmically spaced levels from 1000 to 20 mb (ground-level to about 25 km). For cued networks, 24 hidden-layer neurodes were usually used, while 32 or more were necessary for uncued networks. Despite these large numbers, no network showed any sign of overfitting the data. The retrieval accuracy of the networks was still improving at 48 hidden-layer neurodes, but training times became unreasonably long at larger values. A decreasing schedule of white noise was added to the input patterns, and all networks were trained using momentum. Pruning was not available. For all testing of both temperature and moisture networks, 0.3% white noise was impressed on the network input vector (the simulated sounder output) to simulate the observed noise level of the actual T-1 or T-2 instrument channels.

2.2.2. Moisture. The backpropagation networks for the moisture work were implemented in simulation on a Sun SPARCstation 2 using the commercial package NeuralWorks Professional II/Plus. As before, all results reported here are from networks with a single hidden layer. Because of the need to provide temperature profiles, the input layer of the moisture-retrieval networks contained from 35 to 38 neurodes: five brightness temperatures, 30 temperatures from 1000 to 200 mb, and up to three cues. Because moisture retrievals are only made up to about 200 mb, the Phillips data become erratic above about 300 mb, and the vertical resolution of the T-2 instrument does not justify retrieving moisture concentration at a large number of levels, only the six mandatory levels up to 300 mb were retrieved. Hidden layers having from five to 24 neurodes were investigated. As is often the case in real-world applications, the optimum number lay on a broad, shallow maximum; typically, 11 or 16 hidden neurodes were used for networks trained on data corresponding to a single D-matrix, and 16 or 24 were used for those trained on combined-latitude-band data. Unlike the earlier study, the faster computer and larger data sets allowed virtually every network to overfit the data. Networks were saved at frequent intervals during training, and the one for which recall accuracy, as measured by the average rms output-layer error, was a maximum was chosen for study. As before, white noise was added to the input patterns at the beginning of training and reduced in steps as training progressed. Momentum and pruning were needed on some data sets to optimize recall accuracy.

2.3. Results

2.3.1.Temperature. The closest approximation we could devise to the geographic and other parameters used operationally in so-called D-matrices was to train and test on mixed summer/winter data taken over land within 10°- or 20°-wide bands ranging from 10° S to 60° N latitude. Because only 67 examples were available for most such

---

[8] Weiss, S.H., and C.A. Kulikowski, **Computer Systems that Learn**, 1991, San Mateo, CA: Morgan Kauffman Publishers, Inc.

cases, the retrieval accuracy of networks trained and tested on these subsets were not as good as those obtained operationally. When the networks were trained on the entire over-land training set, however, accuracies comparable to operational ones were obtained. Figure 1 shows retrieval accuracies of cued and uncued networks trained and tested on simulated brightness temperatures representing mixed summer/winter data taken over land and over the entire range of latitudes from 10°S to 60°N. The results are for unfamiliar data; the accuracy measure used is the level-by-level rms difference between the actual and retrieved temperature profiles. The cued network was supplied with all three cues: latitude, hour, and solar zenith angle. The uncued network, although not supplied with these parameters, was implicitly given terrain-height and land/water cues. The retrieval accuracy is very close to that of the cued network.

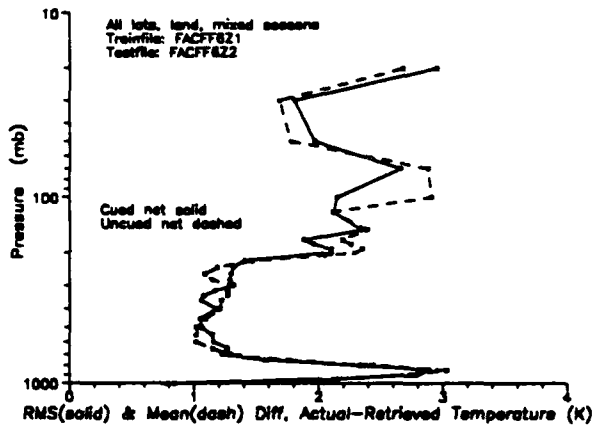


Figure 1. RMS temperature error of cued (solid) and uncued networks trained and tested on all-latitude data.

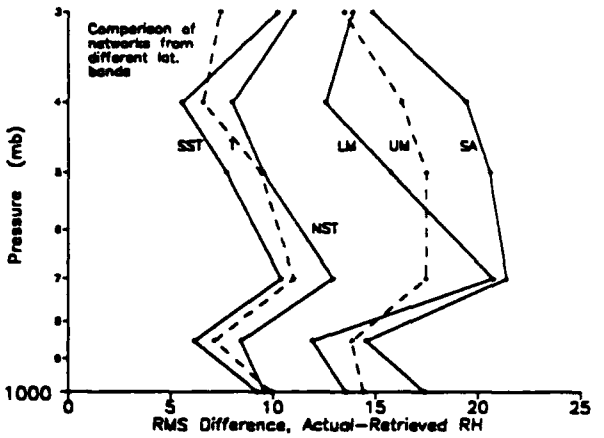


Figure 2. RMS moisture error of cued networks trained and tested on data from individual latitude bands.

2.3.2 Moisture. Figure 2 shows the retrieval accuracies in relative humidity (RH) for cases approximating a single D matrix. The symbols SST, T, NST, LM, UM, and SA indicate, respectively, southern semitropical, tropical, northern semitropical, lower and upper midlatitude, and subarctic. Unless noted, all results in this section represent averages of four networks trained using the four-fold cross-validation method. Figure 3 shows retrieval accuracies of cued and uncued networks trained on data from all latitude bands. They are considerably better than the average accuracy of the networks of Figure 2. As with the temperature retrievals, little or no accuracy is lost if a network is trained and tested on the above combined data but given no geographic or season cues.

Finally, Figure 4 shows the results when the appropriate subarctic testing set was retrieved a the cued network trained on all-latitude data. Clearly, the retrieval accuracy for unfamiliar data corresponding to this single operational matrix presented to a network trained on combined data from all geographic zones and seasons (SA, all) is much better than that obtained from a network trained only on the corresponding single operational-matrix data (SA,SA). The retrieval accuracy of another network trained and tested on all-latitude data is shown for comparison.

2.4. Discussion

The retrieval accuracy of the neural method for both simulated T-1 and T-2 data using no explicit input cues and trained on data from the entire range of seasons and latitudes is comparable to that of operational DMSP temperature-retrieval methods that use restricted latitude ranges and a number of explicit cues. It is possible that the networks were able to find subtle differences among the brightness-temperature input patterns for various latitudes and seasons that

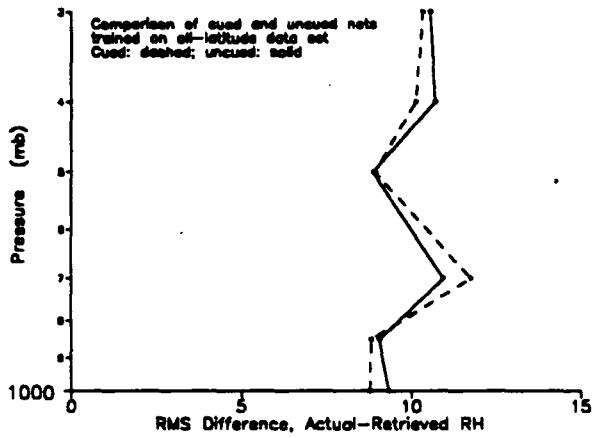


Figure 3. RMS error of cued (dashed) and uncued networks for all-latitude data.

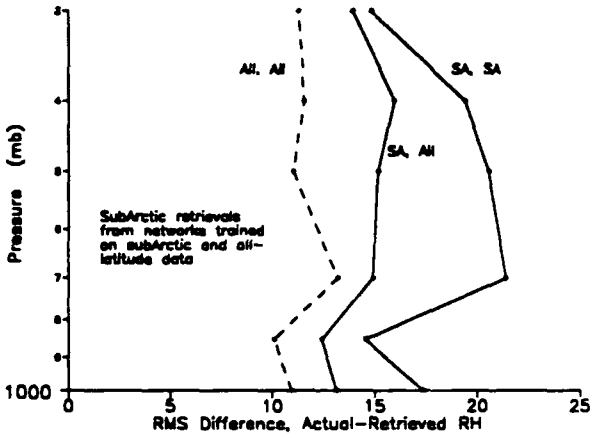


Figure 4. Comparison of subartic RH errors for different training/testing conditions. See text.

allowed them to make accurate models with or without external cues. It is also possible, however, that some of the simplifications required to evaluate the feasibility of the neural retrieval method without confounding influences largely removed the dependence of the data on season or geographic location. We will repeat these measurements with operational T-1 data to answer this and other questions. The most striking result of these studies, however, is the apparent difference between linear and nonlinear methods. Figures 1 and 3 show that networks trained on a very wide range of data — far greater than that possible with the standard statistical method — still can retrieve temperature or moisture profiles with an accuracy at least as good as the traditional statistical method. Moreover, this ability to model nonlinear data is at least partially responsible for the effect seen in Figure 4. During training, a network offered a broad range of data can cross over the artificial boundaries imposed when the data are partitioned into geographic regions and seasons and build a more robust internal model. The results of the simulation studies suggest the eventual possibility of retrieving temperature and moisture profiles over a broad range of latitudes, seasons, and terrain without need of the carefully tuned training data now used in the operational statistical-retrieval method. Even with these encouraging results, however, much work remains before the technique can be truly compared to operational methods.

3. RETRIEVAL STUDIES WITH OPERATIONAL DATA

The primary goal of the current study is to retrieve vertical temperature profiles from operational T-1 data. Secondary goals are to investigate obtaining tropopause and other information normally retrieved by current methods, determine the appropriate number and type of environmental cues, and investigate the behavior of a neural retrieval system under various adverse conditions encountered operationally. Unfortunately, the operational T-1 data were only recently received, and no results are yet available.

4. ACKNOWLEDGMENTS

This work was supported by the Air Force Space Systems Division under the Small Business Innovative Research program. The authors are grateful to Vince Falcone, of the Phillips Laboratory, Geophysics Division, for furnishing a PC version of RADTRAN, to Don Boucher and Bruce Thomas, of The AeroSpace Corporation, for their help and advice throughout the program, and to Joel Susskind, of NASA's Goddard Space flight Laboratory, for his assistance in obtaining and using the Phillips data set.

THIS PAGE IS INTENTIONALLY BLANK

Presented at the Government Neural Network Applications Workshop
24-26 August 1992 GACIAC PR 92-02

Clustering with the Hopfield Neural Networks

Behrooz Kamgar-Parsi and Behzad Kamgar-Parsi
Naval Research Laboratory
Washington, DC 20375

Clustering is an important problem in many fields, hence, many techniques have been developed to solve this problem. In this paper we formulate the unsupervised clustering problem in terms of the Hopfield model of neural nets and compare the results with those of conventional techniques. This neural net approach appears to have two advantages over conventional iterative techniques: When there are well defined clusters it finds good solutions more frequently. When clusters are fuzzy, or when the number of clusters we specify is not compatible with the structure of the data, the neural net indicates that it cannot find valid solutions easily, and that something may be wrong.

1 Introduction

In recent years, clustering techniques inspired by neural networks have been added to the repertory. Kohonen's [6] self-organizing maps have been used to partition the feature space into distinct regions. In this paper we formulate the clustering problem in terms of the *Hopfield model* [4] of neural nets and investigate its performance. The value of neural networks as a computational device is yet to be fully explored. In fact, little is known about the reliability and complexity of these algorithms, and how they scale with the size of the problem. The work we present in this paper is an attempt to answer some of these questions. Hardware implementations of analog neural networks in VLSI are expected to become available in the next few years [7], until then simulation is going to be an indispensible tool in the study and design of these systems. Analog networks are intrinsically synchronous and hence well suited for simulation on massively parallel SIMD machines. By clustering we mean partitioning a set of N patterns (the patterns are represented as points in a d-dimensional metric feature space) into K clusters in such a way that those in a given cluster are more similar to each other than the rest. As there are approximately $\frac{K^N}{K!}$ possible ways of partitioning the patterns among K clusters [1], the problem has exponential complexity and finding the best solution is beyond exhaustive search. Because of the complexity of the problem, finding the best solution may not be possible. This, however, is not a major concern, because in practice usually only *a good* solution is sufficient.

2 Formulation of the Problem

In this paper we use analog neural nets, because they outperform digital nets in solving optimization problems [4, 5]. Many problems of interest, including the problem we address in this paper, can be cast in terms of an energy function, E, that is quadratic in the neuronal activities and has the form [4]

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij}V_iV_j - \sum_{i=1}^{n} I_iV_i + \frac{1}{\tau}\sum_{i=1}^{n}\int^{V_i} dx\, g^{-1}(x)\,. \tag{1}$$

Here n is the number of neurons in the network, and V_i $(0 \le V_i \le 1)$ is the activity or firing rate of neuron i. The first term in (1) is the interaction energy among neurons. In the second term I_i is the bias

or activity threshold of neuron i. The third term encourages the network to operate in the interior of the n dimensional unit cube $\{0 \leq V_i \leq 1\}$ that forms the state space of the system. In this term τ is the self-decay time of the neurons, and $g(u)$, a sigmoid function, is the gain or transfer function of the neurons that relates the input u_i to the output V_i. A standard form for g, which we will also use, is $V_i = g(u_i) = \frac{1}{2}(1 + \tanh\frac{u_i}{u_0}) = \frac{1}{1+e^{-2u_i/u_0}}$,, where u_0 determines the steepness of gain. The neuronal activities, V_i, as well as the input signals, u_i, depend on time t. The evolution of the network is determined by the n coupled ordinary differential equations $du_i/dt = -\partial E/\partial V_i$, which are

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} + \sum_{j=1}^{n} T_{ij}V_j + I_i. \tag{2}$$

We will set $\tau = 1$, so that time is measured in units of τ. To find a solution (i.e. a minimum), we start the network from a randomly selected state and let it evolve until it reaches a minimum of the function E and stops. As is usual in dealing with computationally intractable problems, we find not just one but several solutions by starting the network from different initial states, and then take the best one as *the solution* which may or may not be the optimum. Since a neural network converges rapidly to a minimum we can afford to run it many times thus ensuring that we find at least a very good solution.

We want to partition a set of N points in a 2-D plane (generalization to arbitrary dimensions is trivial) into the *best* K clusters – best in the sense that sum of the squares of the distances of the points from their respective cluster centroids (i.e. sum of "within cluster variances") is minimized. We formulate the problem in a manner that can be solved by a neural network; that is we cast the problem in terms of an energy function that can be minimized by the network.

The energy function will consist of two parts: (i) constraint terms which make certain a point, at the end of the search, belongs to one and only one cluster; (ii) the cost term which is the sum of the residuals and is the function we actually wish to minimize. To partition N points (patterns) among K clusters, we need to have $n = K \times N$ neurons, i.e. K neurons for each point. We denote each neuron by two letters, e.g. neuron pi, and let V_{pi} stands for its activity. The magnitude of V_{pi} represents the weight with which point i belongs to cluster p. The energy function, E, can then be expressed as

$$E = \frac{A}{2}\sum_{i=1}^{N}\sum_{p=1}^{K}\sum_{q\neq p}^{K} V_{pi}V_{qi} + \frac{B}{2}\sum_{i=1}^{N}(\sum_{p=1}^{K} V_{pi} - 1)^2 + C\sum_{p=1}^{K}\sum_{i=1}^{N} R_{pi}V_{pi}^2 \tag{3}$$

where the coefficients A, B and C are positive constants. The *A-term* and the *B-term* are the constraint terms, and together they enforce the *syntax* of the problem. The *C-term* is the *cost* term, where R_{pi}, the square of the distance of point i from the centroid of cluster p (i.e. the residual), is given by $R_{pi} = (x_i - X_p)^2 + (y_i - Y_p)^2$. (x_i, y_i) are the coordinates of point i, and (X_p, Y_p) are the coordinates of the centroid of cluster p. The centroid of cluster p is obtained from the weighted average $X_p = \sum_{i=1}^{N} x_i V_{pi} / \sum_{i=1}^{N} V_{pi}$ and $Y_p = \sum_{i=1}^{N} y_i V_{pi} / \sum_{i=1}^{N} V_{pi}$. Generalizing to higher dimensions is straightforward.

The network dynamics, obtained from $-\partial E/\partial V_{pi}$, are

$$\frac{du_{pi}}{dt} = -u_{pi} - A\sum_{q\neq p}^{K} V_{qi} - B(\sum_{q=1}^{K} V_{qi} - 1) - CR_{pi}V_{pi} + I_{pi}. \tag{4}$$

To find a solution we assign random values between 0 and 1 to all the $n = K \times N$ neuronal activities, V_{pi}, and solve n ordinary differential equations until the activities of all the neurons cease to change.

For simulations we have chosen the following values for the parameters of the energy function: $A = B = 1, C = 0.9/R_{avg}$, all $I_{pi} = 1$, and the gain function parameter $u_0 = 0.1$. Scaling parameter C with the average residual R_{avg} is necessary to ensure good solutions, because as the network evolves, the residuals become generally smaller and the cost term becomes less effective in driving the network toward good solutions; this rescaling of parameter C keeps the cost term of the same order of magnitude as the syntax terms. The parameter values are selected by analyzing dynamical stability of valid solutions [5].

3 Examples

To study the performance of the neural net we have tested it on some examples. In the first data set, there are 128 points divided among 5 clusters with within-cluster Gaussian distributions (Fig. 1a). Here the 5 clusters are rather well defined and out of the 128 trials the neural net found the optimum clusters 128 times. The average number of iterations for convergence was 4263; since $\delta t = 10^{-3}\tau$, the average convergence time is about 4.3τ, where τ is the decay time of a neuron. In VLSI implementations of neural networks that are currently in progress [7], the decay time of neurons, τ, is in the range $10^{-6} - 10^{-3}$ second, hence the convergence time of the network should be in the range of a few microseconds to a few milliseconds.

The conventional method of Forgy [2] in 128 trials found the best clusters only 46 times and various other solutions 82 times. The average number of iterations for convergence was 7. Clearly, in this example, the neural net outperforms the conventional method, in that it finds the best solution much more frequently. On the other hand, the conventional method takes far fewer iterations to converge than the neural net. But we should bear in mind that these are *simulations* of the neural net, and that the number of iterations needed for convergence is not the true measure of the processing time of the network. The convergence time of an actual analog VLSI network must be measured in τ, the characteristic time of a neuron, which is in the micro- to millisecond range.

To test the performance of the network in cases where the clusters are fuzzy, we started from the data points of Fig. 1a, randomly selected 10% of the points and distributed them uniformly throughout the unit square (Fig. 1b). Thus we obtained 5 clusters with uniform background noise. The neural net in 128 trials found the best clusters 28 times. It failed to find valid solutions statisfying the syntax 46 times. This large number of failed solutions can be interpreted as an indication that the clusters are fuzzy, that there are outliers, and that perhaps the specified number of clusters, $K = 5$, is too few. However, even when the syntax is not satisfied we can extract a valid solution with the following scheme. For each point i set the largest V_{pi} to 1 and all the other V_{qi} with $q \neq p$ to 0, and interpret this solution as the one favored by the network; thus we obtain 128 solutions. Conventional algorithms always find valid solutions and cannot give an objective indication of the fuzziness of clusters.

Similarly to Fig. 1b, we generated other data sets by increasing the background noise to 25%, 50%, 75%, and 100% (i.e. no clusters). These data are shown in Fig. 1c-f. The results of partitioning the data among five clusters obtained, in 128 trials, with the neural net and with Forgy's method are listed in Table 2. The average estimated convergence times for the network are given in units of τ. Two points of note in this table are: (i) As the five clusters become less discernible the network increasingly fails to satisfy the syntax, indicating that the clusters are fuzzy and that five clusters are not sufficient. The conventional method, on the other hand, always finds valid solutions, and although the variety of solutions that it finds increases (this is true in both methods), which may be taken as a clue to the fuzziness of the clusters, it is not as objective an indicator as the failure to satisfy the syntax. (ii) When there are well defined clusters the neural net performs better than the conventional techniques, which is reflected in the lower average χ^2 (χ^2 is the sum of within-cluster variances) for the solutions found by the neural net. And as the clusters become fuzzier the quality of the solutions found by both methods becomes comparable.

For the data of Fig. 1f, the centroid trajectories converge to many different points for different trials, which shows that where there is no underlying structure in the data, the network does not prefer any particular clustering and hence finds many different solutions.

To test how this neural net algorithm scales with the size of the problem, we generated 32, 64, 128, and 256 data points, distributed among 5 rather well separated clusters. In all the cases the network found valid solutions in all of its trials, and the best solution in nearly all of its trials; consistently performing better than Forgy's algorithm. Thus, it appears that for clustering the Hopfield network scales well.

In another example we used a standard clustering data set, namely Anderson's 4-dimensional iris data which is composed of three well separated clusters. As was the case with the example of Fig. 1a, the neural net approach found the best solution in all of its 100 trials, whereas the conventional technique found the best solution in 78% of its trials.

Table 1: In this table the results obtained by Forgy's conventional algorithm are compared with those obtained by the neural network. These are based on 128 trials.

Data	Conventional			
	Iter	Best Var	Best%	Avg Var
a	7	0.62	36	1.23
b	8	1.06	34	1.57
c	8	1.95	12	2.27
d	10	2.94	2	3.14
e	10	3.88	10	4.11
f	10	4.46	2	4.64

Data	Neural Net				
	Time	Best Var	Best%	Avg Var	Synt%
a	4	0.62	100	0.62	100
b	7	1.06	22	1.24	64
c	7	1.95	19	2.03	9
d	8	3.00	15	3.04	0
e	6	3.89	1	4.11	1
f	8	4.46	2	4.71	0

Data: refers to the data points of Figs. 1a–f.
Iter: is the average number of iterations for convergence.
Best Var: is the variance of the best solution found.
Best%: is the percentage of trials that found the best solution.
Avg Var: is the average variance of the solutions found.
Time: is the average estimated time of convergence in units of τ.
Synt%: is the percentage of trials that found solutions satisfying the syntax.

References

[1] Feller W (1959) An Introduction to Probability Theory and Its Applications, 2nd edition, Vol 1, p 58. John Wiley.

[2] Forgy EW (1965) Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometric Soc. Meeting, Riverside, California. Abstract in Biometrics 21: 768.

[3] Gear CW (1971) Numerical Initial Value Problems in Ordinary Differential Equations. Prentice-Hall.

[4] Hopfield JJ, Tank DW (1985) Neural computation of decisions in optimization problems, Biological Cybernetics 52: 141.

[5] Kamgar-Parsi B, Kamgar-Parsi B (1987) An efficient model of neural networks for optimization. In: Proceedings of the IEEE First International Conference on Neural Networks, Vol 3, p 785.

[6] Kohonen T (1989) Self-Organization and Associative Memory, 3rd edition. Springer-Verlag.

[7] Mead C (1989) Analog VLSI and Neural Systems. Addison-Wesley.

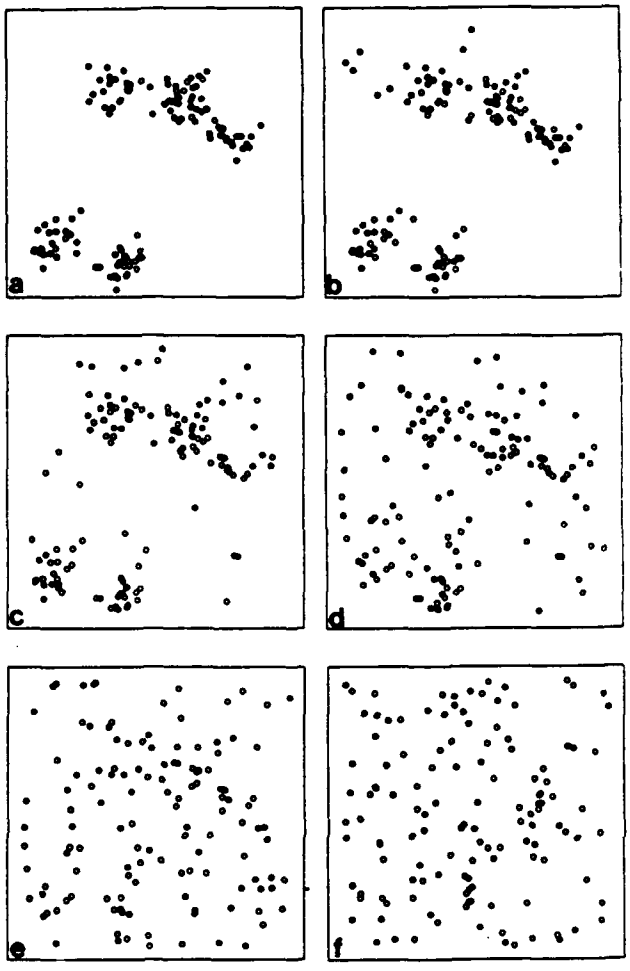


Fig. 1a–f. 128 data points in the unit square: a 5 well defined clusters with within-cluster Gaussian distributions; b same clusters with 10% uniform background noise; c with 25% noise; d 50% noise; e 75% noise; and f uniform distribution or 100% noise

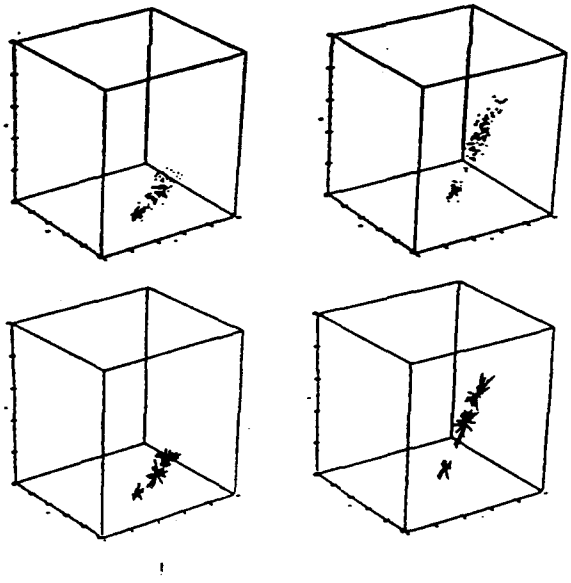


Figure 2. Anderson's 4-dimensional data set composed of 150 samples of three species of iris: (a) and (b) two different 3-dimensional profiles of Anderson's data set. (c) and (d) cluster representation of (a) and (b) profiles, respectively.

THIS PAGE IS INTENTIONALLY BLANK

GACIAC

IIT RESEARCH INSTITUTE 10 WEST 35 STREET, CHICAGO, IL 60616 312/567-4519

ORDER FORM FOR THE PROCEEDINGS OF THE Government Neural Network Applications Workshop - GNN 92

24-26 August 1992

GACIAC PR 92-02

Please complete this form for additional copies of: Volume 1 Unclassified/-Unlimited and Volume 2 Unclassified/Limited (export controlled) proceedings of the workshop above. To receive Volume 2, it is necessary to include your export control number.

The proceedings are published by The Guidance and Control Information Analysis Center (GACIAC), and are available to Government/Military organizations and GACIAC Industrial Subscribers at no charge. Other requestors must submit this completed form with payment of $100.00 for the set, to IIT Research Institute/GACIAC, Attn: Jaime Otero, 10 West 35th Street, Chicago, Illinois 60616-3799.

NAME _____

ORGANIZATION _____

ADDRESS _____

_____

_____

TELEPHONE _____

DTIC USER'S NUMBER _____

EXPORT CONTROL NUMBER _____

THIS PAGE IS INTENTIONALLY BLANK

GACIAC

IIT RESEARCH INSTITUTE 10 WEST 35 STREET, CHICAGO, IL 60616 312/567-4519

TO ORDER GACIAC BULLETIN

The GACIAC Bulletin is published bimonthly and is distributed free-of-charge to Defense Technical Information Center (DTIC) users in the guidance and control community. If you would like to receive a copy of the Bulletin on a regular basis, please complete the following information and mail form to:

IIT Research Institute/GACIAC
ATTN: Jaime Otero
10 West 35th Street
Chicago, IL 60616-3799

DTIC User's Code_____

_____ (____)_____
Name Telephone Number

_____
Company

_____

_____
Address

_____

_____

_____ _____ _____
City State Zip Code

Please distribute a copy of this form to persons in your organization who may be interested in receiving the GACIAC Bulletin.

THIS PAGE IS INTENTIONALLY BLANK

THE TACTICAL WEAPON GUIDANCE AND CONTROL INFORMATION ANALYSIS CENTER (GACIAC)

GACIAC is a DoD Information Analysis Center operated by IIT Research Institute under the technical sponsorship of the Joint Service Guidance and Control Committee with members from OSD, Army, Navy, Air Force, and DARPA. The AMC Smart Weapons Management Office of the U.S. Army Missile Command provides the Contracting Officer's Technical Representative. GACIAC's mission is to assist the tactical weapon guidance and control community by encouraging and facilitating the exchange and dissemination of technical data and information for the purpose of effecting coordination of research, exploratory development, and advanced technology demonstrations. To accomplish this, GACIAC's functions are to:

1. Develop a machine-readable bibliographic data base -- currently containing over 40,000 entries;

2. Collect, review, and store pertinent documents in its field of interest -- the library contains over 14,000 reports;

3. Analyze, appraise, and summarize information and data on selected subjects;

4. Disseminate information through the GACIAC Bulletin, bibliographies, state-of-art summaries, technology assessments, handbooks, special reports, and conferences;

5. Respond to technical inquiries related to tactical weapon guidance and control; and

6. Provide technical and administrative support to the Joint Service Guidance and Control Committee (JSGCC).

The products and services of GACIAC are available to qualified industrial users through a subscription plan or individual sales. Government personnel are eligible for products and services under block funding provided by the Army, Navy, Air Force and DARPA. A written request on government stationery is required to receive all the products as a government subscriber.

Further information regarding GACIAC services, products, participation plan, or additional copies of this proceedings may be obtained by writing or calling: GACIAC, IIT Research Institute, 10 West 35th Street, Chicago, Illinois 60616-3799, Area code 312-567-4526 or 312-567-4519.

JSGCC

JOINT SERVICE GUIDANCE AND CONTROL COMMITTEE